НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 519.688

«До захисту допущено»
Завідувач кафедри СПСКС
_____ Віталій РОМАНКЕВИЧ
(підпис) (ім'я, прізвище)
"___"_____2020р.

Магістерська дисертація

на здобуття ступеня магістра
зі спеціальності 123 Комп'ютерна інженерія

на тему: Спосіб автоматичної корекції дефектів зображень_____

Виконав: студент II курсу, групи КВ-91мп
Місячний Ігор Валерійович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доцент Петрашенко А.В _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю доцент, с.н.с.,к.т.н. Юлія БОЯРІНОВА _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
за освітньо-професійною програмою
Спеціальність 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри СПСКС

_____ Віталій РОМАНКЕВИЧ

(підпис) (ініціали, прізвище)

«1» листопада 2020р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Місячний Ігор Валерійович

(прізвище, ім'я, по батькові)

1. Тема дисертації «Спосіб автоматичної корекції дефектів зображень»,

науковий керівник дисертації к.т.н., доцент Петрашенко А.В,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «12» листопада 2020 р. №3298-С

2. Термін подання студентом дисертації 10 грудня 2020 р.

3. Об’єкт дослідження процес створення та навчання автокодувальників і використання їх для корекції зображень.

4. Предмет дослідження: використання автокодувальників для задач з корекції зображень. Дослідження ефективності використання індексу Соренсена, як метрики для продуктивності нейронної мережі.

5. Перелік завдань, які потрібно розробити: аналіз існуючих методів статичного аналізу зображень; обгрунтування доцільності розробки алгоритму; обгрунтування використання метрик продуктивності нейронної мережі; ПЗ для корекції зображень;

6. Перелік ілюстративного матеріалу - презентація

7. Перелік публікацій - 2 тези. Подання статті на конференцію «ICCSEEA2021: The Fourth International Conference on Computer Science, Engineering and Education Applications»

8. Дата видачі завдання 5 листопада 2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вивчення літератури за тематикою дисертації	10.09.2019	
2.	Підготовка матеріалів першого розділу магістерської дисертації	21.11.2019	
3.	Підготовка матеріалів другого розділу магістерської дисертації	14.02.2020	
4.	Підготовка матеріалів третього розділу магістерської дисертації	19.04.2020	
5.	Підготовка матеріалів четвертого розділу магістерської дисертації	05.05.2020	
6.	Підготовка матеріалів п’ятого розділу магістерської дисертації	30.06.2020	
7.	Розробка програмної бібліотеки	19.07.2020	
8.	Розробка моделі, що демонструє приклад використання програмної бібліотеки	07.09.2020	
9.	Оформлення документації магістерської дисертації	10.10.2020	
10.	Попередній розгляд магістерської дисертації на кафедрі	26.11.2020	

Студент _____ Ігор МІСЯЧНИЙ

(підпис)

Науковий керівник дисертації _____ Андрій ПЕТРАШЕНКО

РЕФЕРАТ

Актуальність теми

Зараз нейронні мережі дуже популярні, кожен день з’являються нові інструменти і дата-сети, за допомогою яких можна вирішувати найрізноманітніші задачі.

Люди дуже активно використовують соціальні мережі та месенджери. Одним із найбільш поширених типів контенту, якими обмінюються користувачі, є фотографія. У зв’язку з цим створюється інструменти, які допомагають оброблювати фотографії, накладання різноманітних ефектів, видалення шуму тощо.

Тому дослідження методів і алгоритмів, які домагають створювати більш ефективні інструменти для редагування зображень є дуже актуальною темою.

Об’єкт дослідження

Процес створення та навчання автокодувальників і використання їх для корекції зображень.

Предметом дослідження

Використання автокодувальників для задач з корекції зображень. Дослідження ефективності використання індексу Соренсена, як метрики для продуктивності нейронної мережі.

Мета роботи

Створення ПЗ, в основі якого лежить використання згорткової нейронної мережі, за допомогою якого можна здійснювати корекцію зображень.

Наукова новизна

1. Проаналізовані існуючі методи та алгоритми статичного аналізу зображень і показано, шо ці методи мають недоліки: мала

контекстуальна кореляція пікселя і його оточення, неможливість аналізу при великих «пустих» зонах.

2. Запропоновано метод для корекцій зображення, який базується на використані нейронних мереж.

Практична цінність

На даний момент існує величезна кількість різних інструментів, яка допомагає користувачам редагувати їх зображення. У більшості випадків, функціональність пов'язана з видаленням об'єктів залишається тільки для досвідчених користувачів. У новачків часто не вистачає досвіду, для того щоб правильно їм скористатися і домогтися свої цілей.

Нейронні мережі мають здатність добре виконувати свої задачі і не брати до уваги досвід користувача. Методи, які описані в даній дипломній роботі, можуть використовуватися для створення додатків, котрі можуть допомагати користувачам редагувати свої фото в одне натиснення кнопки.

Апробація роботи

Основні положення і результати роботи були представленні на:

1. XIII науковій конференції магістрантів та аспірантів "Прикладна математика та комп'ютинг" ПМК-2020 (Київ, 18-20 листопада 2020 р.).
2. VII Міжнародна науково-технічна Internet-конференція «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами». (Київ, 26 листопада 2020 р.).

Структура та обсяг роботи

Магістерська робота складається з вступу, чотирьох розділів на висновків.

У *вступі* подано загальну характеристику роботу, показано наукову новизну і практичну цінність роботи. Зроблено короткий огляд на технологічний стек, який використовується для створення ПЗ.

У *першому* розділі розглянуто і проаналізовано існуючі методи для реставрації зображення. Проведено порівняння класичних статичних методів і використання нейронних мереж в контексті роботи.

У *другому* розділі наведено опис архітектури нейронної мережі та методів її навчання.

У *третьому* розділі наведено особливості практичної реалізації мереж

У *четвертому* розділі наведено результати роботи нейронної мережі та порівняння з класичними методами.

У *висновках* описані результати проведеної роботи.

Ключові слова: Keras, нейронні мережі, згорткові нейронні мережі, машинне навчання, індекс Соренсена.

Зміст

Додатки

Додаток 1. Презентація

Додаток 2. Публікації

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ, ПОЗНАЧЕНЬ, ТЕРМІНІВ

Бібліотека – збірка об'єктів чи підпрограм для вирішення близьких за тематикою задач.

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем.

ОС – операційна система – це базовий комплекс програм, що виконує управління апаратною складовою комп'ютера або віртуальної машини.

ПЗ – програмне забезпечення.

JSON – (JavaScript Object Notation) формат збереження даних.

Бд – База даних

MNIST database – (Mixed National Institute of Standards and Technology database) база даних зображень рукописних цифр.

AE – автокодувальник

ReLU – (Rectified linear unit) випрямлена лінійна функція активації.

ВСТУП

Найпопулярнішими сайтами, якими користуються люди - є соціальні мережі і месенджери. Цілком очевидно, що одним із самих основних типів контенту, якими користуються користувачі, є різного роду зображення. Разом з цим постійно зростає інтерес і популярність інструментів, за допомогою яких можна обробляти фотографії. Функціональність таких інструментів дуже широка. Починаючи від маніпуляцій з розмірами і формами і закінчуючи накладанням фільтрів і різних ефектів.

Реставрувати зображення або видаляти з них об'єкти можна за допомогою алгоритмів, які базуються на аналізі пікселів навколо пошкодженої області. У контексті даної статті, під реставрацією розуміється заповнення кольором місць, які були штучно вилучені з зображення. У випадках з монотонним заповненням тла, наприклад, такі алгоритми справляються відмінно. Проблеми з'являються при реставрації місць, для яких інформації витягнутої з сусідніх пікселів буде недостатньо. Слабке контекстуальное відповідність - це одна із самих основних проблем алгоритмів статичного аналізу пікселів.

Реставрувати зображення або видаляти з них об'єкти можна за допомогою алгоритмів, які базуються на аналізі пікселів навколо пошкодженої області. У контексті даної статті, під реставрацією розуміється заповнення кольором місць, які були штучно вилучені з зображення. У випадках з монотонним заповненням фону, наприклад, такі алгоритми справляються відмінно. Проблеми з'являються при реставрації областей для яких інформації витягнутої з сусідніх пікселів буде недостатньо. Слабка контекстуальна відповідність - це одна із самих основних проблем алгоритмів статичного аналізу пікселів.

Основна ідея полягає в тому, щоб навчити автокодувальники створювати з зображень такий латентий-простір, на основі якого можна згенерувати відреставроване зображення. Варто зауважити, що даний тип нейронних мереж можуть використовуватися для видалення різний дефектів,

наприклад зайвий шум. Проте, в контексті даної статті під реставрацією розуміється видалення штучно створених білих ліній, які нанесенні на оригінальне зображення

РОЗДІЛ 1. ОГЛЯД МЕТОДІВ РЕСТАВРАЦІЇ ЗОБРАЖЕНЬ

В даному розділі розглядаються та аналізуються алгоритми, які відновлюють зображення, проте алгоритми не використовують методи глибокого навчання.

Мета таких алгоритмів полягає у створенні нового зображення, в якому пошкоджена область плавно і безперешкодно об’єднується з іншою частиною зображення. В утвореному зображенні околиці, в яких виконується об’єднання, не повинні мати візуальні артефакти, який побачить типовий глядач. Переважна більшість таких алгоритмів працюють по принципу попіксельного аналізу всього зображення. Тобто, алгоритм шукає певні патерни в зображенні і намагається підставити їх в пошкоджену область. Для вивчення і порівняння було обрано алгоритми, які входять базового набору методів для реставрації зобрадень в популярній бібліотеці OpenCV.

1.1 Дослідження методу Нав’є-Стокса.

Цей метод цифрового малювання намагається відтворити основні прийоми, що використовуються професійними реставраторами. Алгоритм мотивований методом для прямого розв’язання рівнянь Нав’є-Стокса для нестисливою рідини. Функція інтенсивності зображення відіграє роль функції потоку, ізофотні лінії визначають лінії потоку. Після того, як користувач вибере регіони, які необхідно відновлювати, алгоритм автоматично транспортує інформацію в область малювання. Заповнення відбувається таким чином, що ізофотні лінії, що прибувають до меж регіону відновлення, завершуються всередині. Метод, який описаний, не вимагає від користувача явно вказувати регіон зображення, звідки можна скопіювати потрібну частину фото. Це робиться автоматично. Метод не накладає обмежень на топологію регіону, який потребує відновлення, він успадковує

математичну теорію, вже розроблену для рівнянь рідині, включаючи правильну позицію та ефективне використання числових методів.

Основний алгоритм базується на плавності градієнту інтенсивності зображення в напрямку ізофотів:

$$I_t = \nabla^{:\tau} I * \nabla \Delta I,$$

де I – інтенсивність зображення;

$\nabla^{:\tau}$ – градієнт;

Δ - лапсасіан $\partial_x^2 + \partial_y^2$.

Основна умова до якого прагне алгоритм – максимальне приближення до рішення, що задовольняє умові в області фарбування, що ізофоти прямі в напрямку $\nabla^{:\tau} I$ і повинні бути паралельні кривим рівня плавності та інтенсивності зображення, яка при $\vartheta = 0$ дорівнює

$$\nabla^{:\tau} I * \nabla \Delta I = 0$$

1.2 Дослідження Fast marching методу

Роботу Fast marching методу можна представити у вигляді плями масла, яке впало на нерівну поверхню і це пляма поширюється по всій поверхні. Як це пляма буде поширюватися, залежить від кількох факторів, наприклад від гладкості поверхні в кожному місці. Уявіть, що нам цікавий фронт поширення плями, а саме швидкість, за яку фронт досягає кожної точки поверхні. По суті, це і є головна мета методу Fast marching.

Якщо перевести ідею, яка була вказана вище, на математику, то виходить наступне. Розглянемо деяку криву μ, ця крива має розмірність два або більше. Шлях по якому крива поширюється по поверхні залежить від функції швидкості $F(\vec{x})$, яка на ній (на площині) визначена. В алгоритмі нас цікавить як саме крива поширюється по поверхні з плином часу. Для опису положення поширення кривої μ во времени, можна визначити функцію часу

прибуття $T(\vec{x})$, яка проходить через кожну точку $\vec{x}$. Якщо $F(\vec{x}) > 0 \; \forall \; \vec{x}$ на поверхні, тоді ця функція $T(\vec{x})$ задовольняє рівняння Ейконала:

$$|\nabla T|F = 1$$

Ця нотація називається граничної формулюванням. Це нерівність може бути вирішено за допомогою використання Fast marching. Однак якщо $F(\vec{x}) < 0$ в якихось точках $\vec{x}$, в таких випадках цей метод не може бути використаний для вирішення $T(\vec{x})$. Такі ситуації виникають, тому що якась точка може бути переглянуло кривої більш, ніж один раз, отже значення $T(\vec{x})$ може мати кілька значень.

У попередньому параграфі була описана головна мета алгоритму - це знайти функцію $T(\vec{x})$, щоб описати поширення переднього фронту в часі. Для того щоб знайти функцію використовуючи метод, необхідно перетворити вихідну інформацію в дискретне уявлення, в якусь сітку однакового розміру. Функція безперервної швидкості $F(\vec{x})$ стає дискретної функцією, яка визначається в кожній точці сітки з використанням нотації. Для прикладу функція поширення в сітці 4x4. Щоб запустити алгоритм для цього необхідно визначити початковий фронт.

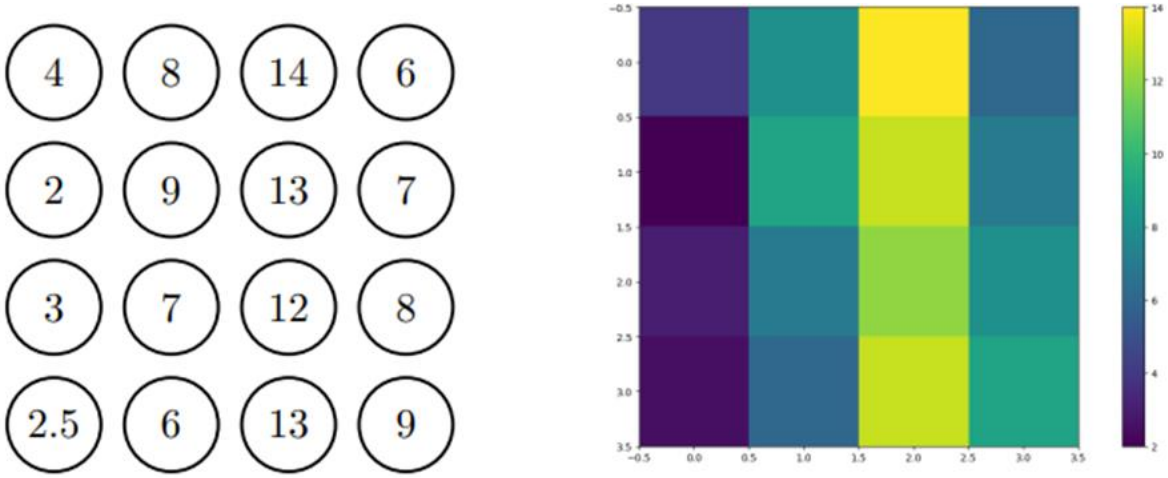


Рисунок 1 – Поширення початкового поширення

Зазвичай це здійснюється за допомогою вибору початкової точки на сітці і визначенні функції $T_{i,j}$ с початкове значення 0. У цілому всі точки на сітці можна розділити на 3 безлічі: far, known, neighbors. Відправною точкою є

елемент known, оскільки $T_{i,j}$ має певне значення. Точки, які знаходяться в безпосередній близькості з початкової точкою, додаються в безліч точок neighbors. Всі інші точки позначаються станом far. Поділ різних точок по трьох згаданих безлічам можна побачити на малюнку 10. Тут метод швидкого переходу вже виконав кілька ітерацій, тому в наборі відомо кілька точок.

Наступним кроком Fast marching методі є калькуляція нового значення $T_{i,j}$ для нових сусідів і оновлення значень, якщо це необхідно.

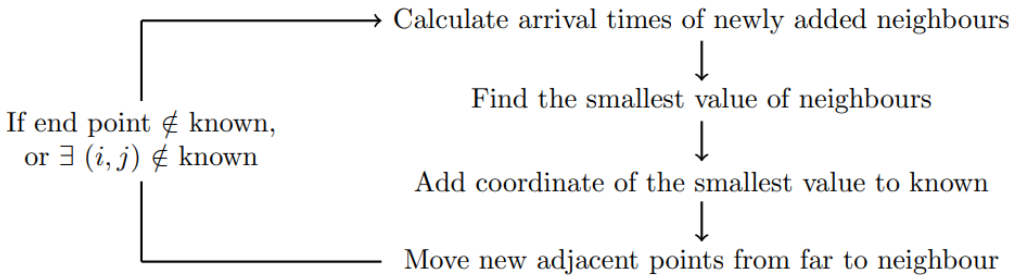


Рисунок 2. Алгоритм роботи методу

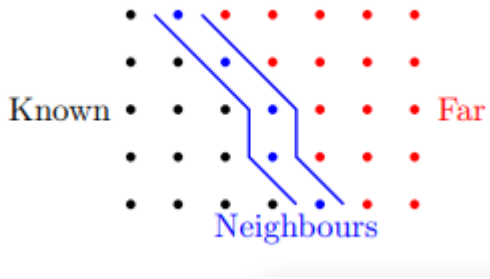


Рисунок 3 – Просування алгоритму вздовж області

1.3 Метод автокодувальників

У цій роботі представлений альтернативний метод реставрації зображення, який грунтується на використанні автокодувальників. Метод зводиться до навчання нейронної мережі таким чином, щоб навчити її видаляти ушкодження з зображення. Основною перевагою такого підходу є поліпшене розуміння контексту навколо області, яка потребує в редагуванні, а так само відсутня необхідність наявності зображення з маскою. Маска - це чорно-біле

зображення, як вказує на область, що редагується. Тобто, нейронна мережа працює повністю автоматично і працює тільки з пошкодженим зображенням.

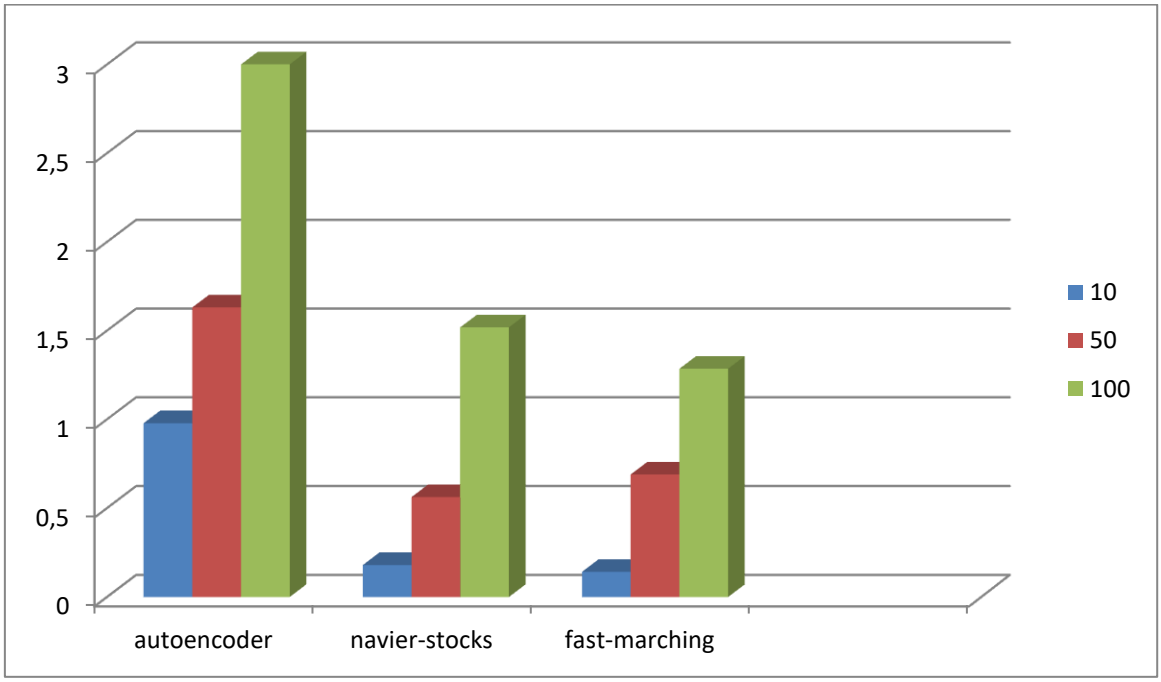
Висновки до розділу 1

Таблиця 1– Порівняння результатів роботи різних методів

Оригінальне зображення	Пошкоджене зображення	Метод Навьє-Стокса	Fast-Marching Метод	Autoencoder
				

Через те, що інформації в сусідніх пікселях недостатньо, то якість зображень, які були створені за допомогою алгоритмів статичного аналізу, значно нижче. Зображення з автокодувальників володіють значно більшою кількістю деталей.

Таблиця 2– Порівняння швидкодії



Тести показують, що автокодувальники володіють найменшою швидкістю обробки. Алгоритми статичного аналізу відмінно оптимізовані під роботу з

зображеннями, тому навіть з різким зростанням навантаження час виконання реставрації зростає лінійно.

Таблиця 3 – Недоліки і переваги методів обробки зображень

Автокодувальнки	
переваги	недоліки
Повністю автоматичний	Втрата інформації
Більш висока деталізація	Нижча продуктивність

Методи статичного аналізу	
переваги	недоліки
Не потребують навчання	Необхідно знати область редагування
Гарно оптимізовані під роботу з зображеннями	Відсутнє розуміння контексту

Грунтуючись на перевагах і недоліках, які наведені вище, можна зробити висновок, що використання автоенкодеров - це не срібна куля, яка вирішує всі проблеми. Підхід має свої плюси і недоліками, як і інші методи для обробки зображень.

2. ОПИС АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ

2.1 Нейронні мережі

Нейронні мережі активно використовуються для вирішення задач, які пов’язані з обробкою зображень. Враховуючи надзвичайну популярність мереж, вже існує велика кількість архітектур і підходів для видалення дефектів. Спроби відтворити здатність біологічної нервової системи навчатися і виправляти помилки привели до створення штучних нейронних мереж. Штучні нейронні мережі являють собою сімейство моделей, побудованих на основі організації і функціонування біологічних нейронних мереж.

На сьогоднішній день нейронні мережі успішно використовуються у вирішні задач з класифікації, каутеризації, машинного зору і тд.

Не дивлячись на наявність відмінностей в архітектурі мереж, всі вони мають спільні риси та властивості. По-перше, в основі кожної нейронної мережі лежить використання досить простих і однотипних елементів, які імітують роботу нейронів головного мозку.

По-друге, спільною рисою для всіх нейронних мереж є здатність до паралельної обробки сигналів, шо досягається за рахунок об’єднання нейронів в шари. На рис.1, представлена структура зв’язків одношарової мережі, де кожен нейрон з попереднього шару пов’язаний з кожним нейроном з наступного шару.

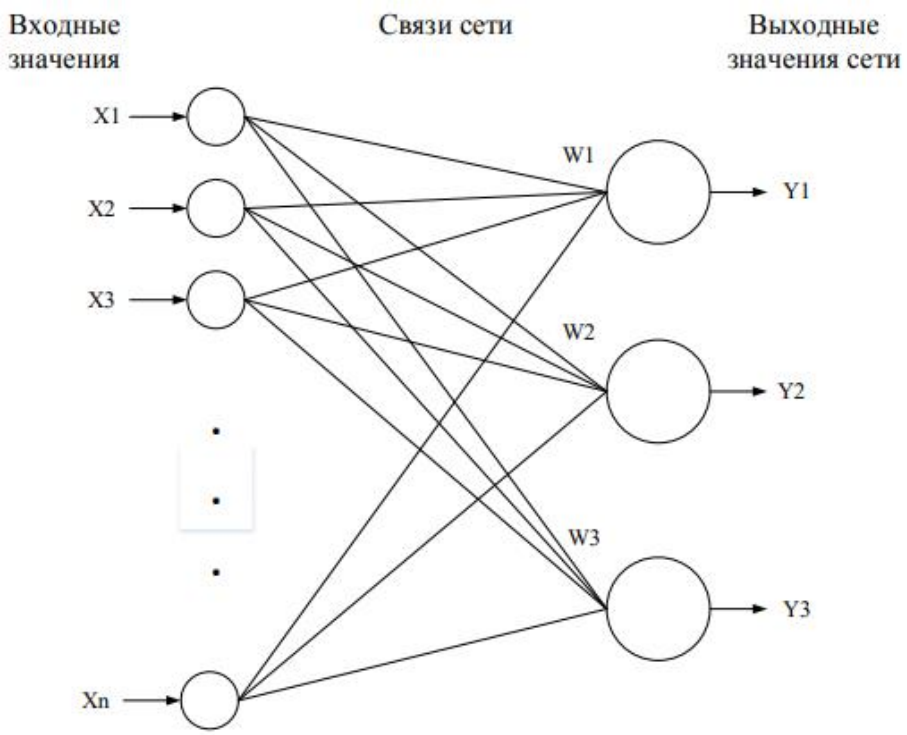


Рисунок 4 – Структура зв’язків в повно зв’язній нейронній мережі

У теорії, кількість шарів і нейронів у шарі може бути вибрано довільним чином, але на практиці їх кількість обмежена обчислювальною потужністю комп’ютерів, на яких мережа тренується і часом, за який мережа повинна проводити обчислення і давати передбачення. Проте, чим складніша структура нейронної мережі, тим масштабніші задачі вона може вирішувати.

Процес функціонування нейронної мережі залежить від величина синаптичних зв'язків (ваг мережі), тому після визначення структури нейронної мережі, для вирішення будь-якої задачі необхідно знайти оптимальні значення всіх параметрів. Цей процес називається навчанням нейронної мережі. Від якості навчання, наскільки правильно будуть підібрані значення параметрів, залежить здатність мережі вирішувати задачі.

2.2 Модель штучного нейрону

В основі будь-якої нейронної мережі лежить модель штучного нейрону, який зображено на рис.2.

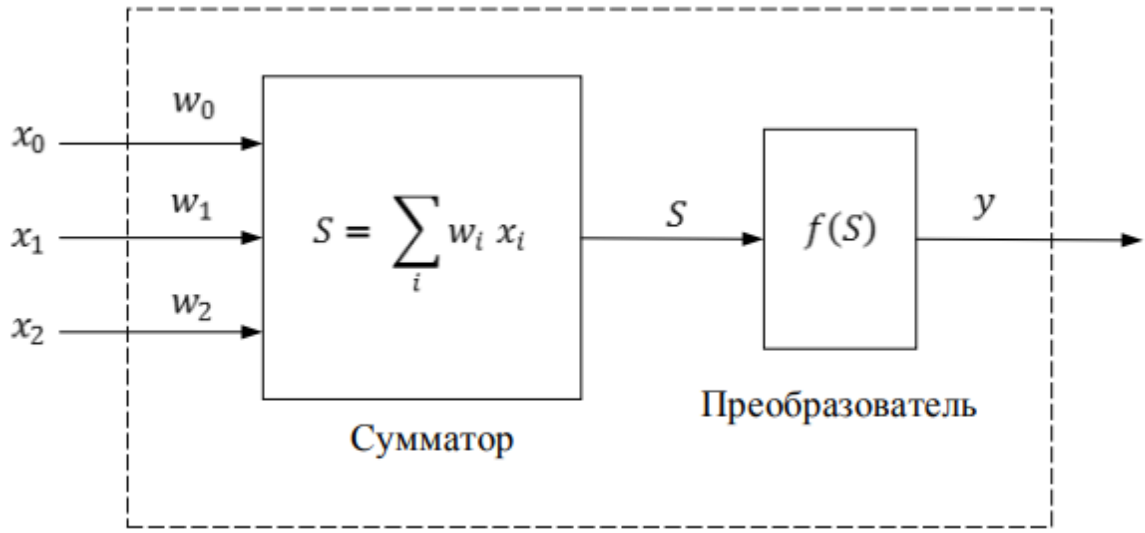


Рисунок 5 – Модель штучного нейрону

У цій моделі нейрон отримує вхідні сигнали (x0…xn), які проходять через синапси. У кожного синапса є певні ваги (w0…wn). Ваги емулюють пропускну здатність синапсів в природних нейронах. Після проходження вхідних сигналів по синапсах, вони оброблюються сумматором. Завдання сумматору – це отримання лінійної комбінації всіх вхідних сигналів, яка служить аргументом для функції-перетворювача. Ця функція визначає значення вихідного сигналу нейрона, яке передається на єдиний вихід – аксон. Таким чином штучні нейрони можуть об'єднуватися в мережі і шари, шляхом з'єднання виходів одним нейронів з входами інших. Отже, штучний нейрон складається з 3 елементів:

1. Синапси та їх ваги, які характеризують силу зв'язку між двома нейронами.
2. Сумматор, який виконує обчислення всіх вхідних сигналів нейрону.
3. Функція-перетворювач. Ця функція називається функцією активації.

Математична модель нейрону описується наступним чином:

$$y = f(S),$$

$$S = \sum_{i=1}^{n} w_i x_i + b.$$

де x – елементи вектору вхідних сигналів;

w – значення ваг зв’язків нейрона;

b – зсув нейрона;

y – вихідний сигнал нейрона.

2.3 Функція активації

Функція активації, яка представлена у формулі 1, визначає вихідний сигнал нейрона.

Взагалі, функція активація може бути будь-якою, але виділяють декілька основних функцій.

1. Порогова передавальна функція. Ця функція описується наступним чином:

$$f\ (S) = \begin{cases} 0, \text{при } S\ < 0 \\ 1, \text{при } S > 0 \end{cases}$$

2. Лінійна функція

$$f\ (S) = \ S$$

3. Лінійна передавальна функція

$$f(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x \geq 1 \\ x, & 0 < x < 1 \end{cases}$$

4. Сигмоїдальна передавальна функція. Сигмоїдальна функція є однією з найбільш поширених функцій, що застосовується в штучних нейронних мережах. Її використання дозволило перейти від бінарних виходів нейрона до аналогових. Бінарність нейронних мереж було зумовлено використанням функцій активації, які приймали значення 0 або 1. Прикладом сигмоїдальної функції може бути логістична функція, яка описується наступним чином:

$$f\ (S) = \frac{1}{1 + \ e^{(-ts)}}$$

де t – це параметр функції, що визначає її крутизну

Коли t прямує до нескінченності, функція починає набувати значення 1. При t = 0 сигмоїда набуває постійного значення 0.5.

5. Випрямлена лінійна функція активації (ReLU). Сигмоїдальна або тангенціальна функція є функціями нелінійними, проте приводять до проблем з загасанням або збільшенням градієнтного спуску в процесі навчання. Функція ReLU на даний момент вважається набагато більш прости і ефективним, з точку зору обчислювальної складності, варіантом функції активації. Функція описується наступним чином

$$f\ (S) = \max(0, S) = \ f(S) = \begin{cases} 0, & S < 0 \\ S, & S \geq 0 \end{cases}$$

На рисунку 3 графік функції ReLU та її першої похідної.

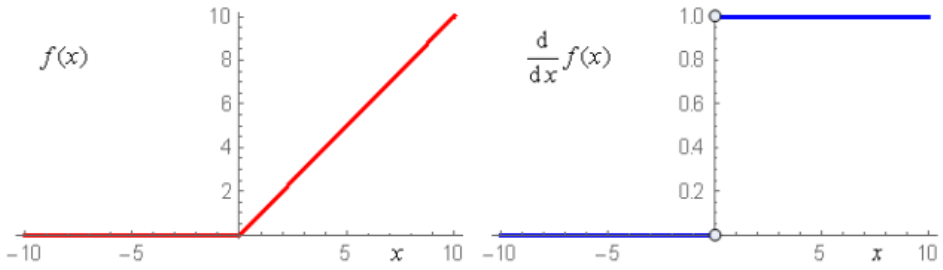


Рисунок 6 – Графік функції ReLU

Її похідна дорівнює 0 або 1, тому не відбувається розростання або згасання градієнтів. На сьогоднішній день існує велика кількість різноманітних модифікацій цієї функції, наприклад Leaky ReLU або Randomized ReLU.

6. Функція Softmax. Ця функція створена для того, щоб перетворити будь-який вектор з реальними значеннями у вектор ймовірностей і розрахувати її можна для *i*-ого нейрону наступним чином:

$$z_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)}$$

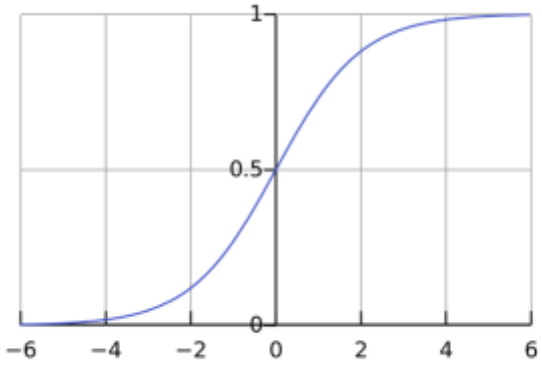


Рисунок 7 – Графік функції Softmax

2.4 Функція втрат

Функція втрат (loss function, cost function) – це метод оцінки того, наскільки коректно алгоритми прогнозуються правильний результат. Таким чином, нейронні мережі навчаються за допомогою функції втрат. Для різних задач, сценаріїв, алгоритмів та наборів даних потрібні різні функції втрат. У машинному навчанні не існує універсальної функції, яку можна використовувати для будь-якої задачі. Можна виділити декілька найрозповсюдженіших функцій.

1. Середньоквадратична помилка

   Середньоквадратична помилка – це середнє значення квадрату різниці між передбаченням та фактичним значенням. Функція визначена наступним чином:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

   де y – значення передбачення, яке генерується мережею;

   $\hat{y}$ – фактичне значення;

   n – кількість тренувальних прикладів

2. Абсолютна помилка

   Подібно до середньоквадратичної помилки, середня абсолютна помилка стосується лише середньої величини помилки і не враховує напрямок похибки. Однак абсолютна помилка є більш стійкою для ризької флуктуації значень, оскільки вона не підносить значення в квадрат.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

2.5 Алгоритм навчання нейронної мережі

Найбільш поширеним алгоритмом навчання нейронних мереж є метод градієнтного спуску. Даний алгоритм відноситься до методів навчання з учителем.

В процесі навчання нейронної мережі із застосуванням алгоритму зворотного поширення помилки, на вхід подається велика кількість прикладів з навчальної вибірки. Один цикл, при якому було використанні всі приклади називають епохою. Процес навчання відбувається від епохи до епохи до тих пір, поки ваги і значення зміщень не стабілізуються, а похибка мережі на зійдеться до певного мінімального значення. Алгоритм зворотного поширення помилки може бути реалізований 3 способами:

Послідовний режим. Послідовний режим так же відомий як стохастичний градієнтний спуск. В цьому режимі зміна ваг зв’язків відбувається після подачі кожного прикладу з навчальної вибірки.

Пакетний режим. У пакетному режимі навчання, процес налаштування ваг відбувається після подачі всіх прикладів, тобто після кожної епохи.

Mini-batch**.** Між двома способами, які було перераховано вище, існує певний компроміс. У цьому випадку коригування ваг налаштування ваг відбувається після невеликої кількості навчальних зразків.

З точку зору ефективності навчання, послідовний режим є більш ефективним. Така перевага зумовлена тим, що при корекції ваг після кожної епохи робить процес навчання більш стохастичним. Таким чином зменшується вірогідність зупинки алгоритму у точці локального мінімуму функції.

Алгоритм роботи методу зворотного поширення помилки для пакетного режиму навчання мережі. Даний метод навчання нейронної мережі також називають правилом Error backpropagation. Він був запропонований в 1986 р. Румельхарктом та Маккелендом.

Цей алгоритм використовується для мінімізації відхилення від реальних значень вихідний значень від тих до яких прагне мережа. Для прикладу в якості функції, яка визначає продуктивність мережі будемо використовувати метод найменших квадратів:

$$E(w) = \frac{1}{2}\sum_{i,k}(f_{i.k} - y_{i,k}^{t})^2$$

де $f_{i,k}$ – значення вихідного сигналу вихідного *k*-го нейрону мережі, при подачі на її вхід *i*-го набору тренувальних даних,

$y_{i,k}^{t}$ – необхідне значення вихідного сигналу *k*-го нейрону мережі, при подачі на її вхід *i*-го набору тренувальних даних.

У якості функції мінімізації функції $E(w)$ використовується метод градієнтного спуску, який забезпечує корекцію вагових коефіцієнтів наступним чином:

$$\Delta w_{ij}^{q} = \mu\frac{\partial E}{\partial w_{ij}}$$

де Δw_{ij}^{q} – величина зміни ваги зв'язку, що з'єднує *i*-й нейрон *q-1* шару с *j*-м нейроном шару *q*;

μ – коефіцієнт швидкості навчання.

Таким чином, вага зв’язку змінюється пропорційно її вкладу в значення помилки нейрона. Зміна ваги зв’язку визначається наступним чином:

$$\Delta w_{ij}^{q} = \mu \delta_i x_i$$

де δ_i – значення помилки *j*-го нейрону в шарі q

x_i – значення *i*-го вхідного сигналу для *j*-го нейрону в шарі q.

Значення помилки нейрону визначається в залежності від його положення в мережі. Для нейронів вихідного шару:

$$\delta_j = (f_{i,k}(S))' \, (f_{i,k} - y_{i.k})$$

де $y_{i.k}$ – значення до якого прагне мережа;

$f_{i,k}$ – фактичне значення вихідного сигналу k-го нейрону для *i*-ї епохи,;

$(f_{i,k}(S))'$ - значення похідної функції активації сигналу k-го нейрону для *i*-ї епохи. Якщо нейрон належить одному з прихованих шарів, тоді

$$\delta_i^q = (f_i^q(S))' \sum w_{ij} \delta_i^{q+1}$$

де δ_i^q – помилка *i*-го нейрона в шарі q;

δ_i^{q+1} – помилка *j*-го нейрону в $q + 1$ шарі;

w_{ij} – вага зв’язку, яка з’єднує нейрони;

$(f_{i,k}(S))'$ - значення похідної функції активації сигналу k-го нейрону в шарі q.

Із цього можна зробити висновок, що вплив кожного нейрону на величину помилок нейронів в наступному шарі, пропорційне значенню помилки цього нейрону.

2.6 Згорткові нейронні мережі

З появою великих обсягів даних і великих обчислювальних можливостей широкою популярності набули нейроні мережі, особливу популярність здобули згорткові нейронні мережі.

Згорткова нейронна мережа (CNN) – архітектура штучних нейронних мереж, яка була запропонована Яном Лекуном, призначена для ефективного розпізнавання зображень.

Свою назву згорткові мережі отримали через присутність операції згортки, суть якої полягає в обчисленні нового значення поточного пікселя, враховуючи значення сусідніх пікселів. Для обчислення нового значення використовується ядро згортки. Ядро накладається на поточний і сусідні елементи матриці. Далі підраховується сума, де складовими є добуток значень пікселів на значення елемента ядра, яка накрила піксель. Одержаний результат підсумовується і зберігається в аналогічній позиції вихідного зображення. Процес згортки зводиться до проходження ядром згортки по початковій матриці і отриманню матриці меншої розмірності.

В архітектуру мережі закладені апріорні знання з предметної області комп’ютерного зору:

1. Піксель зображення сильніше пов’язаний з сусіднім (локальна кореляція)
2. Об’єкт на зображенні може зустрітися в будь-який інший частині зображення.

2.7 Архітектура згорткової нейронної мережі

Архітектура згорткової нейронної мережі представляє чергування згортальних шарів, повнозв’язних та агрегувальних шарів. Всі види шарів можуть розташовуватися в мережі в довільному порядку.

У свою чергу, шар, який утворюється після згортки, утворюється за рахунок того, що зображення попереднього шару сканується невеликим вікном і пропускається крізь набір ваг. Таким чином кожен нейрон виконує згортку деякої області попереднього шару. Отже, набір згорток – це карти особливостей вхідного зображення і кожна згортка знаходить характеристик, які притаманні тільки попередньому шару.

На рисунку 5 зображена стандартна структура згорткової нейронної мережі.

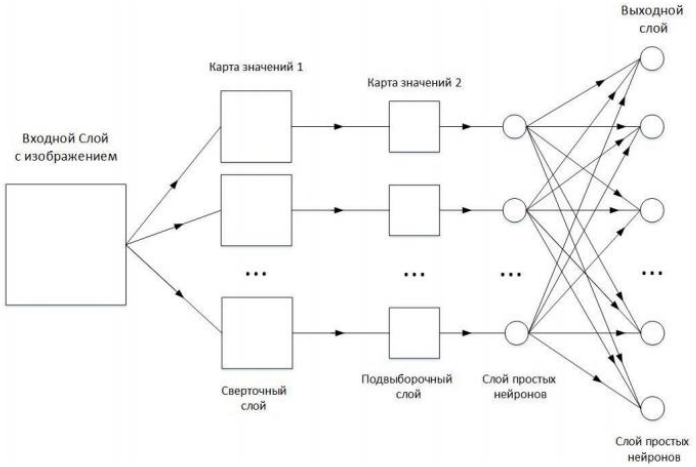


Рисунок 8 – Структура згороткової нейронної мережі

2.8 Згортковий шар

Кожен нейрон, який знаходиться в площині згорткового шару, отримує свої вхідні значення від відповідної області попереднього шару. Тобто, вхідне зображення проглядається невеликим вікном і пропускається крізь набір ваг, отриманий результат записується в відповідний нейрон згорткового шару.

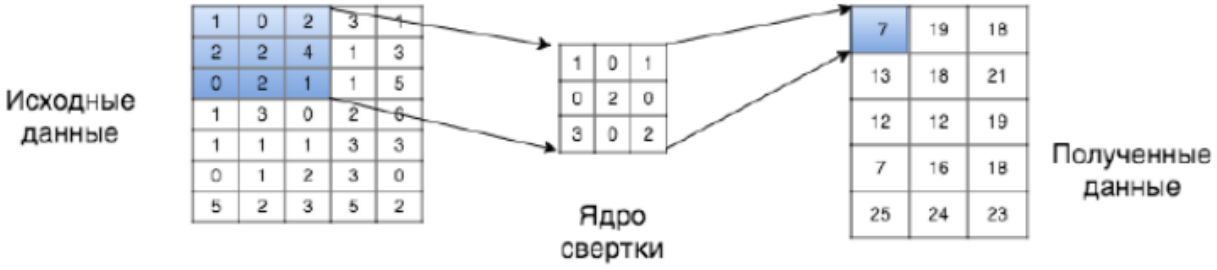


Рисунок 7 – Принцип роботи згорткового шару

Таким чином, процес функціонування нейрона згорткового шару $y_k^{(i,j)}$ описується виразом:

$$y_k^{(i,j)} = b_k + \sum_{s=1}^{k} \sum_{t=1}^{k} w_{k,s,t}^{x^{((i=1)+(j+t))}},$$

де $y_k^{(i,j)}$ – нейрон k-ий нейрон площини згорткового шару;

b_k – зміщення нейрону k-ої площини;

k – розмір рецептивної області нейрону;

$W_{k,s,t}$ – матриця синоптичний коефіцієнтів, x – виходи нейронів попереднього шару.

Отриманий результат згортки кожної локальної області пропускається через функцію активації:

$$out_k^{(i,j)} = f(y_k^{i,j})$$

Згортковий нейронний шар зменшує розмірність вихідного шару.

2.9 Подвиборочний шар

В сучасних нейронних мережах використовується субдискретизуючі шари, які зменшують розмірність вхідної карти ознак. Це можна робити різними способами, але найчастіше для цього використовується метод вибору максимального елементу в околиці поточного елемента. Вся карта ознак поділяється на комірки з яких вибирається найбільше по значенню число. Використання такого підходу дозволяє зробити мережу інваріантною до масштабних перетворень.

Рисунок 9 – Принцип роботи нейронів подвиборочний шару

Зазвичай цей шар йде відразу за згортковим і має таку саму топологію, шо і попередньому згортковому шарі.

2.10 Повнозв’язний шар

Шар, в якому кожен нейрон з’єднаний з усіма нейронами на попередньому рівні, кожен зв’язок має свій ваговий коефіцієнт.

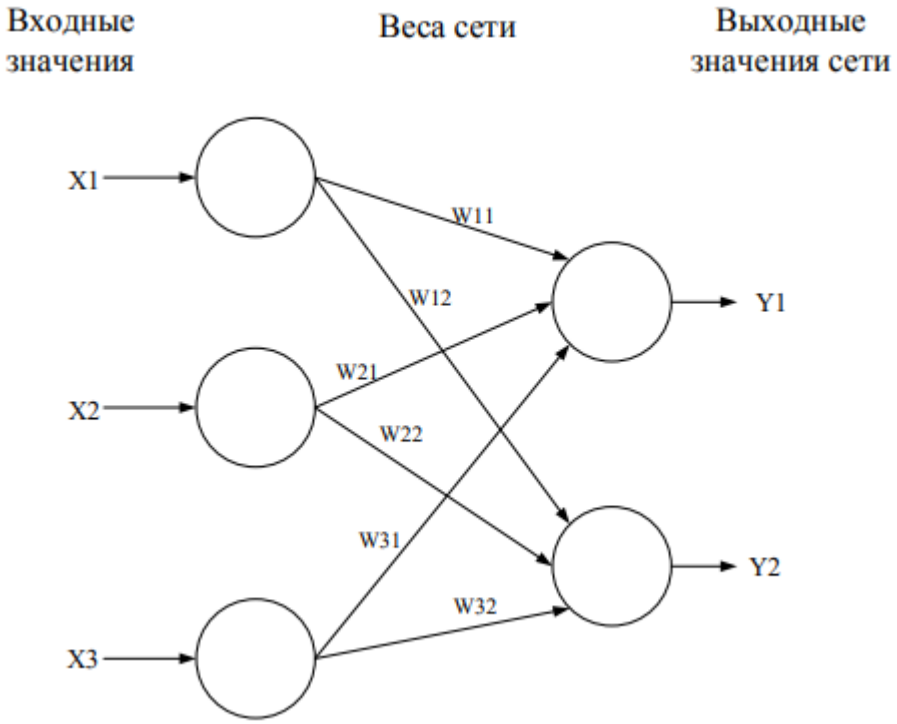


Рисунок 10 – Повнозв’язний шар

DropOut є простим і ефективним методом регуляризації. Метод полягає, що в процесі навчання мережі з її топології багаторазово випадковим чином виділяється підмережа, і чергове оновлення ваг відбувається тільки в рамках виділеної підмережі. Кожен нейрон видаляється з поточної мережі з певною ймовірністю, яка називається коефіцієнтом dropout.

2.11 Переваги та недоліки згорткових нейронних мереж

Переваги:

- Гарно вирішує задачі з класифікації і розпізнавання зображень
- Має меншу кількість параметрів, які налаштовується в порівнянні з повнозв’язною нейронною мережею.
- Порівнюючи з повнозв’язними мережами (наприклад парсептроном) має більш високу стійкість до зміни положення зображення, яке оброблюється.

До недоліків згорткових нейронних мереж можна віднести велику кількість гіперпараметрів, до яких належать:

- Кількість згорткових і підвоборочних шарів.
- Кількість карт ознак в згортальних і підвоборочних шарах.
- Розмірність матриці згортки.
- Функція активації нейронів.
- Швидкість навчання мережі

Всі вище перераховані параметри істотно впливають на результат роботи нейронної мережі, вони як правило підбираються емпірично. Для популярних задач існують вже готові налаштування мереж.

2.12 Автокодувальники

Автокодувальник – це нейронна мережа, яка копіює вхідні дані на вихід. За архітектурою схожа на персептрон. Автокодувальники стискают вхідні дані для подання їх і latent-space (прихований простір), а потім відновлюють за цього подання вихідні дані. Мета – це отримати у вихідному шарі найближчий до вхідних даних результат.

Відміна особливість автокодувальників – кількість нейронів на вході та виході збігається.

Класична реалізація автокодувальників складається за 3 частин:

1. Енкодер: відповідає за стиснення вхідного зображення в latent-space. Представлений функцією кодування *h = f(x);*
2. Latent-space: латентний простір, який містить в собі набір особливостей, які енкодер знайшов у вхідному зображенні.
3. Декодер: призначений для відтворення вхідних даних з латентного-простору. Представлений функцією декодування *h = f(x).*

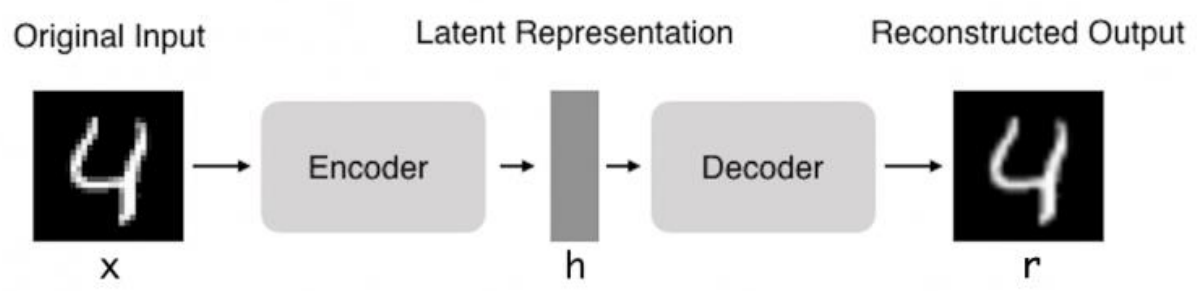


Рисунок 11 – Структура мережі

Таким чином атокодувальник описують функцією *g(f(x)) = r*, де *r* збігається з початковим значенням *x* на вході.

Якби єдиним завдання автокодувалинків було копіювання вхідних даних на вихід, їх цінність можна було б поставити під питання. Розробники мережі розраховували на те, що приховане уявлення h буде володіти корисними властивостями.

Цього домагаються створенням обмежень для задачі копіювання. Один із способів отримати корисні результати в автоенкодера – обмежити *h* розмірами меншими, ніж *x*.

Автокодувальники – це алгоритм стиснення даних з парою важливих властивостей:

1. Автокодувальники мають змогу ефективно стискати дані тільки з однієї вибірки, тобто дані повинні мати подібний набір характеристик. Така властивість робить неможливим використання таких нейронних мереж

замість інших алгоритмів стиснення даних, як gzip, наприклад. Тобто, якщо мережа навчена працювати з зображенням котів, то ефективне стиснення зображень с автомобілями неможливо.

2. Втрати. Вихідна множина автокодувальника завжди буде відрізнятися від того, що подається на вхід. Представлення буде дуже схожим, але завжди буде присутня деградація якості. Якщо є необхідність використовувати алгоритми без втрат, то автокодувальники точно не підходять для вирішення таких задач.

3. Без вчителя. Для процесу навчання мережі непотрібно розмічати початкові дані. Можна просто використовувати «сирі» дані. Такі мережі самі знаходять та виводять з даних маркери.

Автокодувальник може вирішувати задачі копіювання, не витягуючи корисної інформації про розподіл даних, якщо

- Розмірність прихованого уявлення збігається з розмірністю входу;
- Розмірність прихованого уявлення більше, ніж розмірність входу;
- У видках, коли дата-сет для навчання дуже великий.

У цих випадках навіть лінійний енкодер і лінійний декодер копіюють вхідні дані на вихід, не вивчаючи нічого корисного про розподіл. В ідеалі можна організувати будь-яку архітектуру автокоенкодера, задаючи розмір коду, енкодера і декодера, в залежності від складності розподілу, яку необхідно моделювати.

Функція втрат визначається за формулою:

$$l\big(f(x)\big) = \frac{1}{2}\sum_k(\hat{x}_k - x_k)^2,$$

де $\hat{x}_k$ – результат який генерується мережею;

x_k – значення, до яких мережа пране;

l(f(x)) – лінійна функція активації.

2.13 Використання автокодувальників

Два основних практичних застосування автокодувальників для візуалізації даних:

- Згладжування шуму
- Зменшення розмірності

З відповідними обмеженнями по розмірності і розрідженості, автокодувальники можуть вивчати data projections, які більш цікаві, ніж PCA (метод головних компонент) або інші базові техніки.

Автокодувальники навчаються автоматично на дата-сетах. Це значить, що легко натренувати частини алгоритму, які згодом будуть добре працювати на конкретній топології даних і немає необхідності в застосуванні нової техніки, а тільки відповідні дані для навчання.

Однак автокодувальники будуть погано справлятися зі стисненням зображень. По мірі того, як автокодувальник навчається по заданому дата-сету, він досягає розумних результатів стиснення аналогічних тим даним, які використовується для навчання, но з’являються проблеми, у випадках використання його у якості компресора загального призначення. Наприклад, стиснення JPEG компресія буде набагато краще.

Автокодувальники навчені, як зберігати інформацію, так і надавати нові властивості. Для цього використовують декілька типів автокодувальників. Проте завжди існує спір між зміщенням і дисперсією. З одного боку гарно мати декодер, який гарно вміє відновлювати вихідні дані. З іншого боку хочеться зробити латентний простір мінімальним.

Типи авктодувальників:

- Класична реалізація автокодувальника
- Багатошаровий автокодувальник
- Роздріджений автокодувальник
- Згортковий автокодувальник

- Авктодувальники для видалення шуму

Для цієї роботи використовується згортковий автокодувальник. Це зумовлено тим, що енкодер і декодер є згортковими нейронними мережами.

2.14 Роздріджений автокодувальник

Одним із способів, щоб впоратися с компромісом, який описано вище, є посилення розрідженості прихованих активацій. Це може бути додано поверх вузького місця, або замість нього. Існує дві стратегії для забезпечення регуляризації розрідженості. Вони схожі на звичайні регуляризацію, де вони застосовуються на активаціях замість ваг. Перший спосіб – це застосувати регуляризацію L1, яка викликає розрідженість

$$L_1 = \sum_i (y_i - y(t_i)^2) + \alpha \sum_i a_i^2$$

де a_i є активацією на і-му прихованому шарі.

Іншим способом є використання KL-дивергенції, яка є мірою відстані між двома розподілами ймовірності. Замість того, щоб налаштовувати параметр α , можна припустити активацію кожного акту нейрона, як змінну Берулі з ймовірністю p і налаштовувати цю ймовірність. У кожній партії навчання вимірюється фактична ймовірність і різниця обчислюється та застосовується, як фактор регуляції. Для кожного нейрону *j* розраховується емпірична ймовірність

$$\hat{p} = \frac{1}{m} \sum_i a_i(x)$$

де *i* – ітерується по всім зразкам в партії.

2.15 Автокодувальники для видалення шуму

Автокодувальники для видалення шуму можна розглядати або як варіант регуляризації або як надійні автокодувальники, які використовуються

для виправлення помилок. У цих архітектурах вхід порушується деяким шумом (наприклад, адитивний білий гаусів шум або використання шару Dropout) і очікується, що нейронна мережа відновить чисту версію.

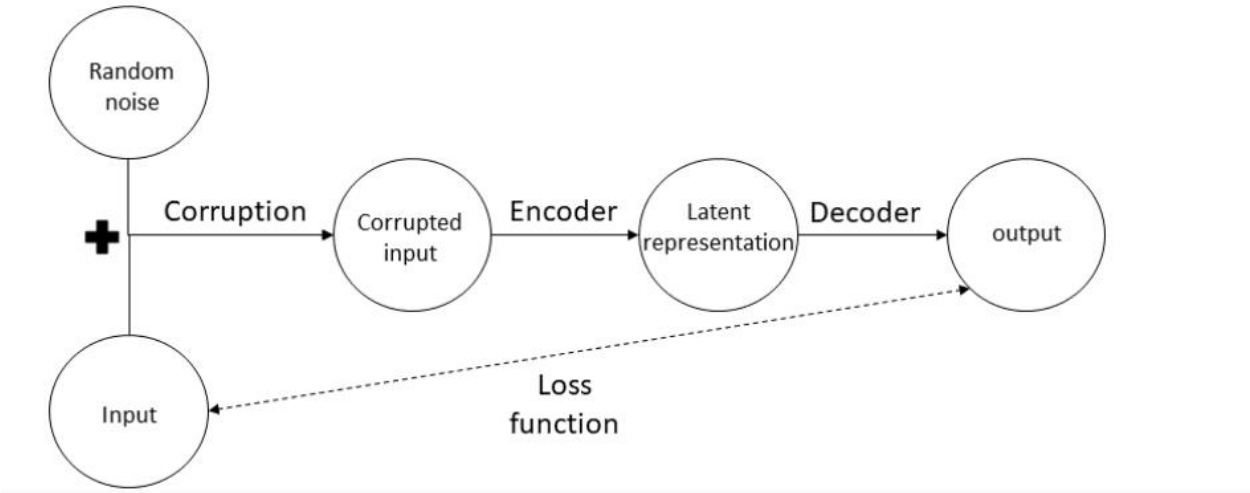


Рисунок 12 – Структура автокодувальника для видалення шуму

Рисунок схема роботи автокодувальника для видалення шуму

Нехай $\tilde{x}$ – є випадковою величиною, розподіл якої задається через

$$C_p(\tilde{x}|x) = N(x, \sigma^2 I)$$

або

$$C_p(\tilde{x}|x) = \beta * x,$$

де “*” множення елементів. В першому випадку варіаційний параметр σ визначає силу шуму. В іншому, параметр p встановлює ймовірність того, що значення x не буде зведене нанівець.

2.16 Контрактні автокодувальники

У створенні автокодувальників для мушу акцент робиться на тому, щоб дозволити кодеру бути стійким до деяких сигналів. У контрактих автокодувальниках акцент робиться на розробку функції, яка знаходить властивості в даних, таким чином, щоб вона ставала менш чутливою до

невеликих відхилень. В наслідок цього, змушуємо енкодер ігнорувати зміни у вхідних даних, які не є важливими для реконструкції декодером. Матриця Якобіана прихованого шару h складається з похідної кожного вузла h_j відносно кожного значення x_j на вході x. Функція реконструкції втрат і втрата регуляризації насправді призводять до протилежних результатів. Мінімізуючи квадрат норми кожний латентий простір стає схожим на інші простори, таким чином відновлення зображень стає більш складною задачею, тому що дуже маленька різниця між відображеннями. Основна ідея в полягає в тому, щоб видалити всі непотрібні деталі з латентного-простору, внаслідок цього зменшити регуляризацію.

2.17 Варіаційні автокодувальники

Суттєвим покращенням можливостей представлення атокодувальників було досягнуто за допомогою варіаційних автокодувальників (VAE). Слідом за варіаціним Байєсом (VB) – генеративні моделі, які намагаються описати генерацію за допомогою імовірнісного розподілу. Зокрема, враховуючи набір даних, який використовуються $X = \{x_i\}^N$. Для кожного елементу з вибірки, ми припускаємо генеративну модель для кожного x_i. Це зумовлено невизначеною випадковою прихованою змінною z_i, де параметри регулюють генеративний розподіл. Ця генеративна модель також еквівалентна до імовірнісного декодування. Симетрично, можна припустити приблизний розподіл над прихованою змінною z_i, яка отримана завдяки вхідному x_i.

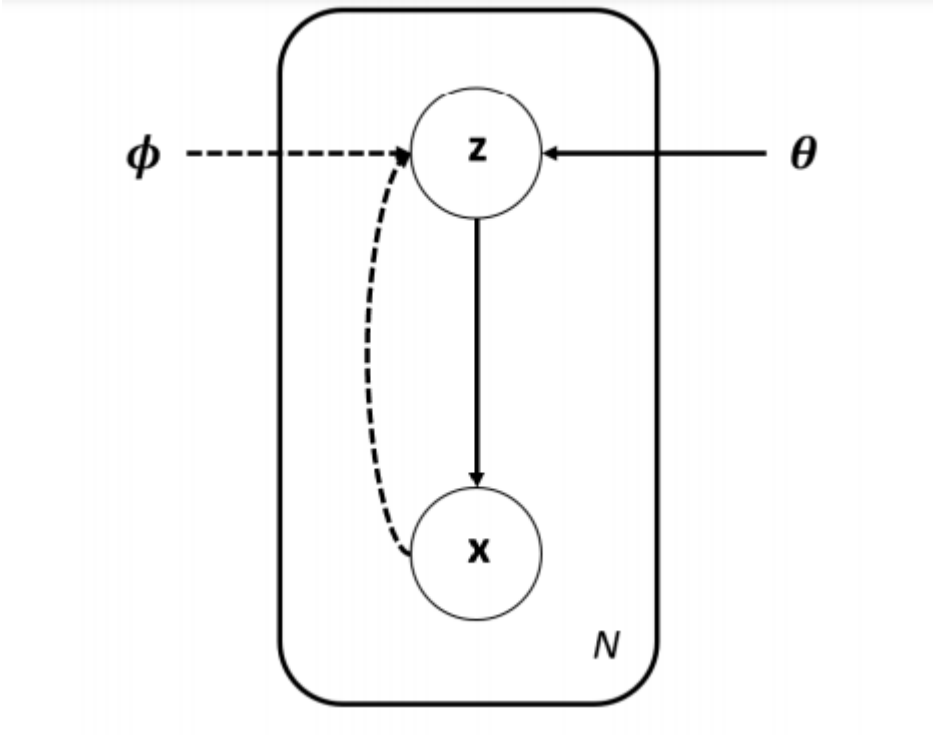
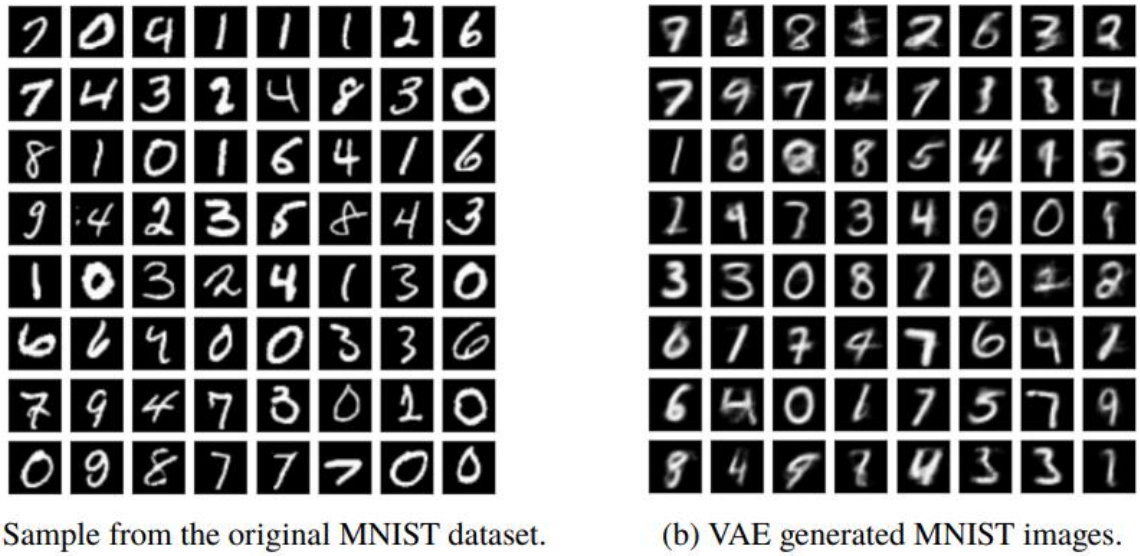


Рисунок 13 – структура VAE

2.18 Використання автокодувальників як генеративної моделі



а) Sample from the original MNIST dataset. (b) VAE generated MNIST images.

Рисунок 14 – Приклад згенерованих зображень

Варіаційні автокодувальники – це генеративні моделі, які намагаютьсяописати генерацію даних за допомогою імовірнісного розподілу. Як було зазначено вище, розподіл, який є похідною для енкодера,

регулюються в напрямку безперервного і повного розподілу в форму заздалегідь визначеного попередника прихованих змінних. Обговорюючи питання генерації нових зразків, то безпосередня дискусія передбачає порівняння між VAE та GAN. Перший тренується на MSE, яке дає трохи розмиті зображення але дозволяє робити висновок над прихованими змінними, щоб контролювати вихід.

2.19 Структура нейронної мережі

Як було зазначено вище структура автоенкодера складається з 3 частин. Енкодер і декодер являють собою згорткові нейронні мережі. Між цими двома мережами знаходиться латентне-простір, який містить в собі всю необхідну інформацію для відновлення зображень.

В роботі використовується одна із найпопулярніших архітектур згорткових нейронних мереж – U-net. U-Net вважається однією з стандартних архітектур CNN для задач сегментації зображень, коли потрібно не тільки визначити клас зображення цілком, але і сегментувати його області за класом. Архітектура складається з стискуючої частини, яка захоплює контекст. І семеричній їй розстягуючої частини, яка дозволяє здійснити точну локалізацію. Мережа навчається наскрізним способом на невеликій кількості зображень і перевершує попередній метод, який використовує ковзаюче вікно. Використовуючи ту ж мережу, яка була навчена на зображеннях світлової мікроскопії пропускання (фазовий контраст і DIC), U-Net посіла перше місце в конкурсі ISBI 2015 року з трекінгу клітин в цих категоріях з великим відривом. Крім того, ця мережа працює швидко. Сегментація зображення 512 на 512 пікселів займає менше секунди на сучасному графічному процесорі.

Для U-Net характерні такі ознаки:

- Досягнення високих результатів в різних реальних задачах, особливо в біомедичних додатках.

- Мережа показує досить гарні результати, навіть при невеликому датасеті.

2.20 U-net архітектура

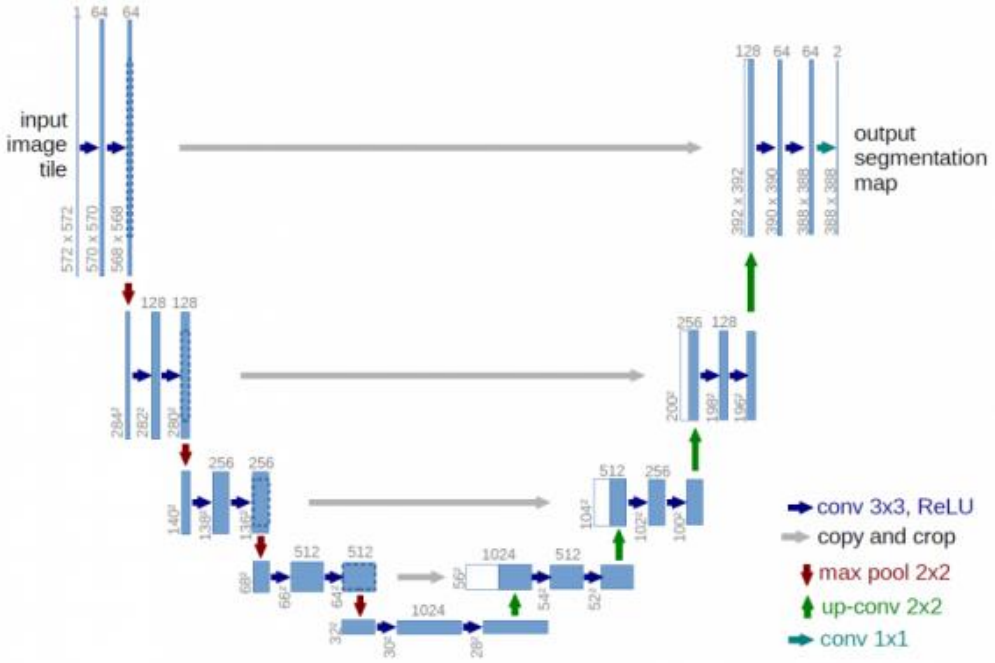


Рисунок 15 – Архітектура U-net (приклад зображення з розширенням 32 на 32 пікселі). Кожний синій квадрат відповідає багатоканальній карті властивостей. Розмірність карт приведена над відповідним квадратом.

Звужувальна частина – це типова архітектура згорткової нейронної мережі. Вона складається з повторного використання двох згорток 3 на 3, після яких використовується функція ReLU і операція максимального об’єднання для зниження розмірності зображення. На кожному етапі понижувальної дискретизації розмірність карт властивостей подвоюються. Кожен крок у розширювальному шляху складається з операції підвищення дискретизації карт властивостей, за якою слідують:

- Згортка 2 на 2, яка зменшує розмірність карти властивостей.
- Дві згортки, за якими слідує ReLU.

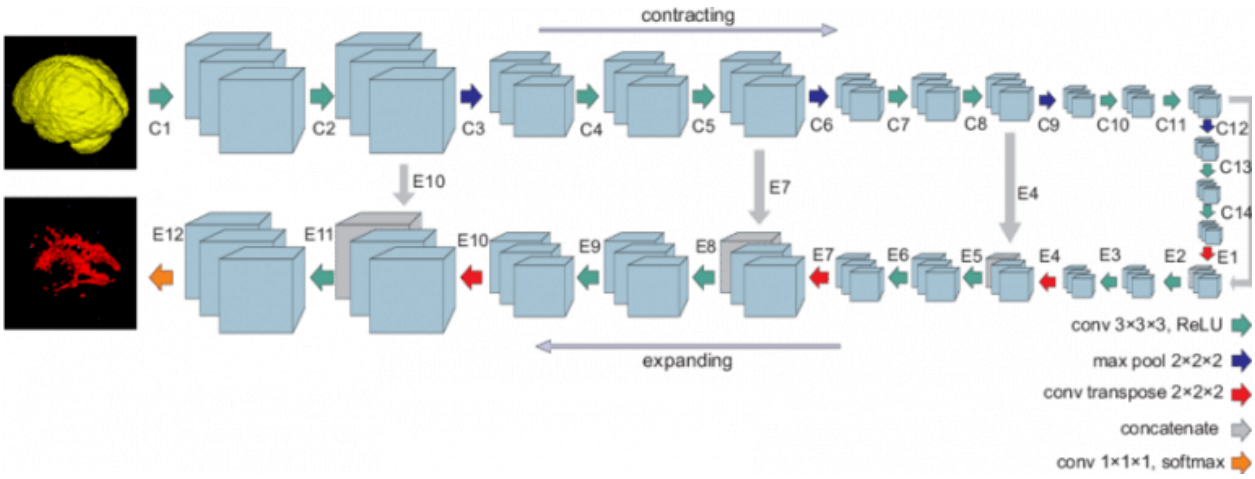


Рисунок 16 – Схема роботи U-net мережі

На останньому шарі використовується згортка 1 на 1 для зіставлення кожного 64-компонентного вектора властивостей з бажаною кількістю класів. Всього мережа містить 23 згорткових шари.

2.21 Навчання U-net мережі

Мережа навчається методом стохастичного градієнтного спуску на основі вхідних зображень і відповідним їм карт сегментації. Через згорки вихідне зображення має меншу розмірність Процес навчання мережі зводиться до того, щоб навчити енкодер генерувати краще простір і зберігати тільки необхідні властивості. Помилка грунтується на різниці двох зображень, різниця між відновленим зображенням і оригінальним зображенням з дата-сету.

З ростом кількості епох зростає і якість роботи енкодера і декодера. Можна як змінюються відреставровані картинки, які виходить за підсумками роботи мережі. Результати навчання зображено на рисунку 17. Так само варто звернути увагу на зростання індексу Соренсена. Індекс відображає схожість вихідного зображення і відреставрованого.

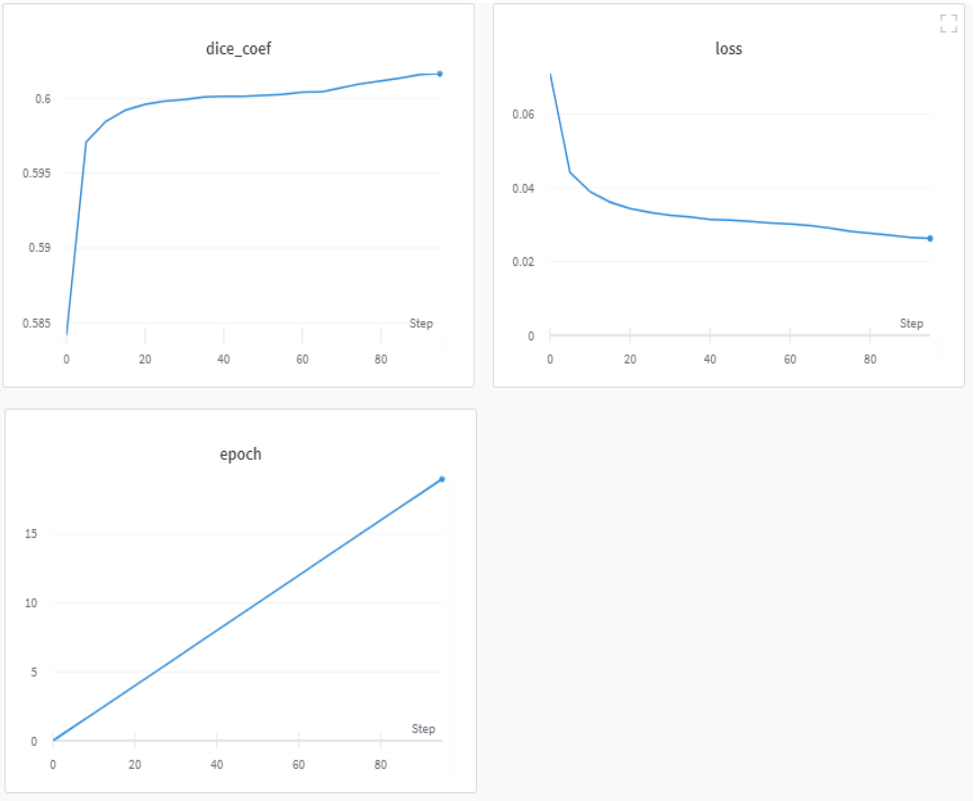
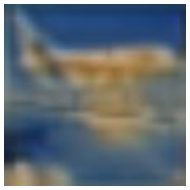
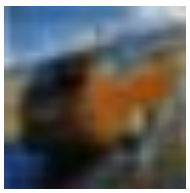
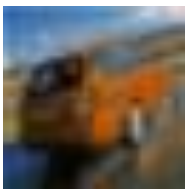
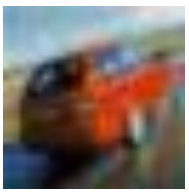
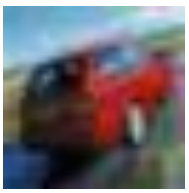


Рисунок 17 – Графік прогресу навчання нейронної мережі

5 епоха	10 епоха	15 епоха	20 епоха
			
			
			

Таблиця 5 – еволюція результатів роботи мережі

2.22 Метрики продуктивності нейронної мережі

Маючі метрики продуктивності, розробник має можливість робити висновки з приводу покращень, які вносяться до кодової бази проекту. Наприклад, якщо після внесення змін на графіках спостерігається падіння показників продуктивності, то зміни можна прибрати і ніяк не впливати на працездатність мережі. Маючі правильні і гарно підібрані метрики, розробник має можливість керувати процесом оптимізації, а не робити їх наосліп.

В контексті даної роботи навчання нейронної мережі проходить за допомогою звірення двох зображень – пошкодженого зображення та зображення з «target data». Такий принцип навчання активно використовується при рішення іншої популярної задачі «контекстуальна сегментація». Практично пошук відповідних Далі приведені метрики, які використовуються при навчанні моделей.

2.23 Піксельна відповідність

Точність пікселів, майбуть концептуально найпростіша для розуміння. Це відсоток пікселів у зображенні, які були правильно класифіковані. Хоч це легко зрозуміти, найчастіше це не найкраща метрика.

На перший погляд може бути важко розгледіти проблему цієї метрики. Наприклад через нейронну мережу пропускається зображення, яке розташоване зліва (Рис 2). Зображення справа – це правдиве зображення до якого прагне мережа. У цьому прикладі нейронна мережа намагається сегментувати кораблі на супутникову знімку. Нехай точність нейронної мережі 95% і вона згенерувала наступне зображення (Рис 4). Результат, який зображено на рисунку, складно назвати очікуваним. Отже, якщо модель класифікує як цей клас, 95% пікселів класифікуються точно, тоді як інші 5% -

ні. Навіть якщо мережа правильно знаходить кораблі на фото, то її точність сегментації не може перевищувати 5%.

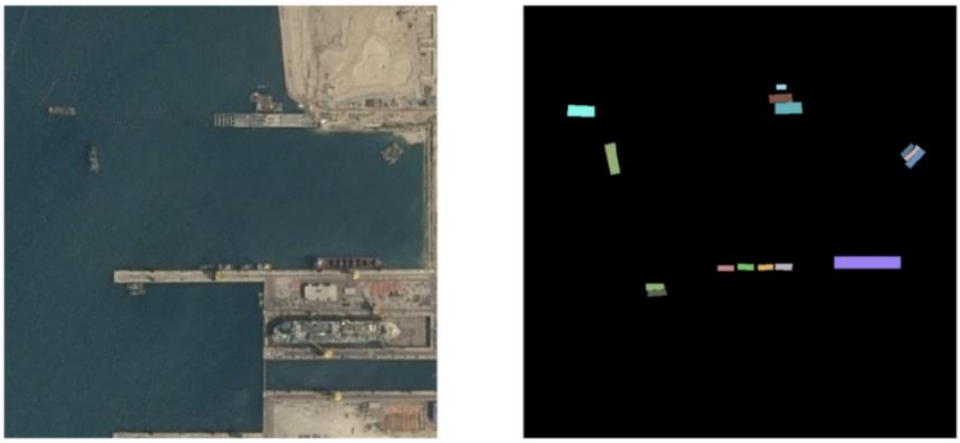


Рисунок 18 – Приклад семантичної сегментації

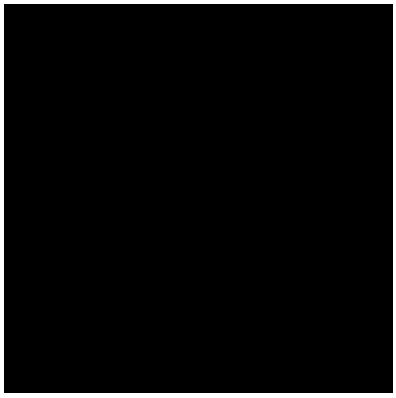


Рисунок 19 – Приклад згенерованого зображення

Це питання називається дисбалансом класів. Коли класи на зображенні незбалансовані, це означає, що певний клас буде завжди займати більшу частину зображення, ніж інші класи. На жаль, дисбаланс класів поширений у багатьох реальних дата-сетах, тому його не можна ігнорувати. Нижче будуть представлені метрики, які краще вирішують цю проблему.

2.24 Intersection-Over-Union (IoU)

Метрика також відома як Індекс Жакарда, є однією з найбільш часто використовуваних метрик у семантичній сегментації. IoU є дуже простою метрикою, яка надзвичайно ефективна.

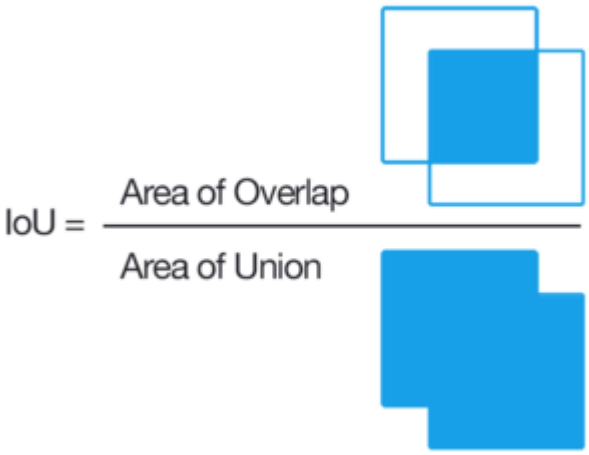


Рисунок 20 – Візуалізація IoU метрики

IoU – це область перетину між істиною та передбачуваною сегментацією, як показано на зображені. Ця метрика коливається в межах 0 – 1, при цьому 0 означає відсутність перекриття та 1 – ідеально перекривається.

Для бінарної сегментації (два класи) або багатокласової сегментації середнє значення IoU зображення обчислюється шляхом обчислення індексу для кожного класу їх усереднення.

Для демонстрації переваг такого методу, можно скористатися тим самим прикладом з сегментацією кораблів (Рис. 3). Нехай розміри зображення становлять 100 пікселів. Для підрахунку перекриття можна уявити, що передбачене зображення розташовуємо над правдим зображенням. Після розташування перевіряються пікселі корабля, які перекриваються. Оскільки не існує пікселів, які класифікуються як кораблі, тому за формулою перетин дорівнює 0. Об’єднання складається з усіх пікселів, класифікованих як кораблі з обох зображень. У цьому випадку існує 5 пікселів (це довільний вибір числа), які класифікуються як загальна

кількість кораблів. При діленні перекриття на перетин, отримаємо результат, що становить лише 47.5%.

2.25 Dice Coefficient / Sorenson Index

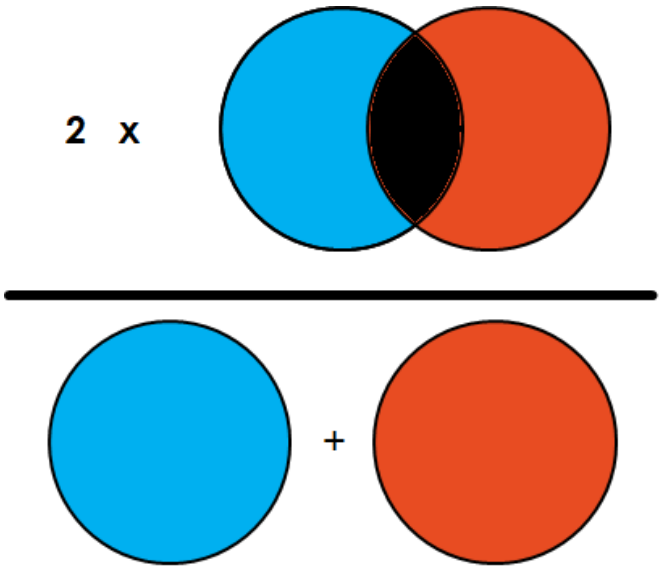


Рисунок 21 – Ілюстрація, як обчислюється індекс

По приблизно схожому алгоритму, який було описано вище, обчислюється і Індекс Соренсена. Наприклад,

Загальна кількість пікселів на двох зображеннях – 200;

Область зображень, яке перетинається: 0;

$$\frac{2 * \text{Область перетину}}{\text{загальна кількість пікселів}} = 0;$$

Область зображень, яке перетинається = 95

$$\frac{2 * \text{Область перетину}}{\text{загальна кількість пікселів}} = 0 = 95*2/200 = 0.95$$

Dice = (Ships + Background)/2 = (0%+95%)/2 = 47.5%

У цьому випадку було отримано однакові значення, як для методу IoU так і для індексу Соренсена. Варто зазначити, що результати не завжди будуть рівні. Коефіцієнт Dice дуже схожий на IoU. Вони позитивно корелюють,

тобто, якщо один каже, що модель А краща за модель В при сегментації зображення, то інший скаже те саме. Як і IoU, вони обидва коливаються від 0 до 1, причому 1 означає найбільшу подібність між передбачуваним та істиною. Проте, як правило, індекс Соренсена показує більш правдиві результати.

Висновки до розділу 2

В даному розділі було проаналізовані методи для вимірювання продуктивності навчання мережі. Беручі до уваги те, що викладено вище, було обрано індекс Соренсена. Ця метрика активно використовується для запровадження більш якісного навчання мережі.

Проведено аналіз та порівняння існуючих згорткових нейронних мереж, які можна використовувати як основу для розробки автокодувальників.

На основі проведеного аналізу представлено метод на базі використання згорткових нейронних мереж, які можна використовувати для вирішення задач з реставрації зображень.

РОЗДІЛ 3. ПРОЕКТУВАННЯ І РЕАЛІЗАЦІЯ

3.1 Проектування і реалізація нейронної мережі.

В якості мови програмування, яка використовувалась для створення нейронної мережі, було обрано Python. Основною причиною такого вибору є високий рівень базових абстракцій і гарну сумісність з різними фреймворками машинного навчання. Дата-сет для навчання CIFAR10.

В якості навчальної вибірки було обрано дата-сет CIFAR10, який має 60 тисяч різноманітних кольорових зображень. Зображення зберігаються у форматі 32x32 пікселів і класифіковані, як 10 окремих класів. У вибірці міститься фото літаків, машин, птахів, котів, оленів, жаб, кораблів, вантажівок, собак і коней. Кожен клас має по 6 тисяч зображень.

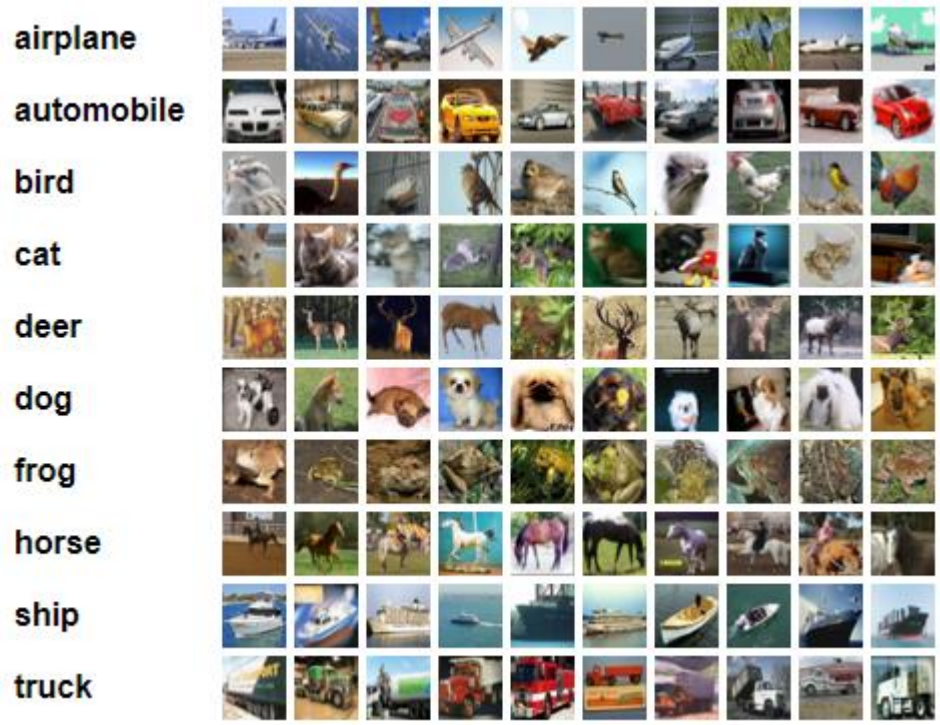


Рисунок 22 – Набір зображення з CIFAR10

3.2 Формування власного дата сету з пошкодженими зображеннями

В контексті даної роботи для навчання використовуються два дата-сети. Перший – це зображення, які містяться в CIFAR10. Інша вибірка для навчання формувалася власноруч. В його основі лежить використання вищеописаного вибірки і використання популярної бібліотеки для роботи з зображеннями OpenCV. В другій вибірці з зображенням міститься штучно зіпсовані зображення. Варто зазначити, що під зіпсованим зображенням розуміється зображення на яке випадковим чином нанесені білі лінії, які в процесі реставрації необхідно видаляти.

Фактично створення зіпсованих зображень можна розбити на три кроки.

1. Випадкове обрання зображення з CIFAR10.
2. Створення за допомогою OpenCV масок. Маски – це білі лінії, які були штучно згенеровано.
3. Нанесення маски на оригінальне зображення.

OpenCV – це бібліотека функцій та алгоритмів комп'ютерного зору, чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки і аналізу вмісту зображень. Нижче представлено програмний код мовою Python, який забезпечує генерацію пошкоджених зображень. Функціональність коду наступна

`write_to_file i read_from_file` – це допоміжні функції, які надають інтерфейс для запису та читання файлів.

`create_corrupted_image` – функція, яка створює маску і накладає її на зображення. Алгоритм роботи функції наступний:

1. Генерація пустого зображення, яке заповнене білим кольором. Розмір маски повинен дорівнювати розміру зображень з основного дата-сету. В нашому випадку 32 на 32 пікселі.

2. Генерація випадкового числа від 1 до 10. Згенероване число визначає кількість білих ліній, які будуть в масці.
3. Для кожної лінії генеруються її координати.
4. Випадково визначається товщина ліній.
5. За допомогою використання API, яке надається при використання OpenCV, створюється чорні лінію і накладаються на заздалегідь підготовлене біле зображення.
6. Накладання маски на оригінальне зображення.
7. Зберігання результатів на локальний диск.

Програмний код, який генерує зображення

```
import cv2
from PIL import Image
import numpy as np

def create_corrupted_image(img):
  ## Prepare masking matrix
  mask = np.full((32,32,3), 255, np.uint8)
  for _ in range(np.random.randint(1, 10)):
    # Get random x locations to start line
    x1, x2 = np.random.randint(1, 32), np.random.randint(1, 32)
    # Get random y locations to start line
    y1, y2 = np.random.randint(1, 32), np.random.randint(1, 32)
    # Get random thickness of the line drawn
    thickness = np.random.randint(1, 3)
    # Draw black line on the white mask
    cv2.line(mask,(x1,y1),(x2,y2),(1,1,1),thickness)

  # Perforn bitwise and operation to mak the image
  masked_image = cv2.bitwise_and(img, mask)

  return masked_image

def write_to_file(image, dst="dst.png"):
  cv2.imwrite(dst, image)

def read_from_file(path):
  cv2.imread(path)


imageFromCIFAR10 = read_from_file('./cifar10/dog/image1.jpg')
result = create_corrupted_image(imageFromCIFAR10, mask)
write_to_file(result)
```

Рисунок 23 – Результат накладання маски на оригінальне зображення

3.3 Підбір архітектури згорткових нейронних мереж

Велика кількість навчальних і тестових прикладів вимагає досить великого часу навчання, тому, для підбору найкращої конфігурації мережі було вирішено скористатися готовими бібліотекам. Для підбору найбільш вдалих параметрів мережі, був проведений огляд існуючих бібліотек машинного навчання.

Існує безліч програмних засобів для вирішення завдань машинного навчання, в таблиці 2 наведені найбільш популярні з них, вони надають можливості для створення повнозв'язних нейронних мереж (FC), згорткових нейронних мереж (CNN) та автокодувальників (AE).

Назва	Мова	ОС	FC NN	AE	CNN
DeepLearnToolbox	Matlab	Windows, Linux	+	+	+
Pylearn2	Python	Linux	+	+	+
Theano	Python	Windows, Linux,Mac	+	+	+
Darch	R	Windows, Linux	+	-	+
Torch	Lua, C	Linux, Mac	+	+	+
Caffe	C++	Linux, OS X	+	+	-

Tensorflow	C++, Python, JS	Linux, OS X, windows	+	+	+
Cuda CNN	Matlab	Linux	+	+	-

Беручі до уваги порівняльну таблицю, для роботи для створення нейронної мережі було обрано бібліотеку з відкритим сирцевим кодом Tensorflow. Розробники підтримали можливість її використання майже на буль-якій платформі і в більшості популярних мов програмування.

Keras – це бібліотека, що дозволяє на більш високому рівні працювати з нейронними мережами. Вона спрощує вирішення безлічі задач, використовується для розробки MVP і сильно зменшує кількість одноманітного коду. В якості бібліотеки, яка використовується всередині Keras, було налаштовано tensorflow.

3.4 Створення нейронної мережі

В контексті даної роботи використовуються автокодувальник, енкодери і декодери яких використовуються згорткові нейронні мережі. Для будування таких мереж Keras має зручний програмний інтерфейс, який за декілька рядків коду реалізує необхідно функціональність.

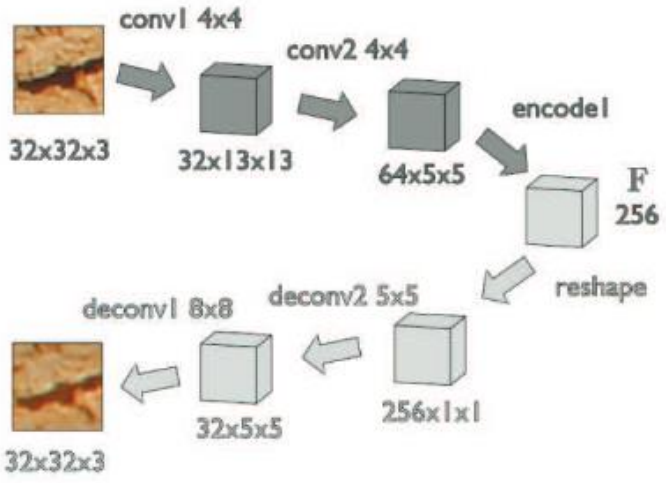


Рисунок 24 – Архітектура згорткових автокодувальників

Шар	Розмір на виході з шару	Фільтр	Крок (stride)
Вхідний сигнал	3 x 32 x 32		
Конволюційний	8 x 13 x 13	4 x 4	2
Конволюційний	16 x 5 x 5	4 x 4	2
Повнозв'язковий	F		
Деконволюційний	8 x 5 x 5	5 x 5	2
Деконволюційний	1 x 32 x 32	8 x 8	5

```
input_img = keras.Input(shape=(32, 32, 1))

x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = layers.MaxPooling2D((2, 2), padding='same')(x)
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = layers.MaxPooling2D((2, 2), padding='same')(x)
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = layers.MaxPooling2D((2, 2), padding='same')(x)
```

Реалізація енкодера

В коді, який представлено вище, створюється згорткова нейронна мережа. Головна задача мережі – це створення латентного-простору, яка використовується для подальшої генерації. Аналізуючи код можна спостерігати, як утворюється латентий простір, відбувається стискання зображень з 32x32 до зображення 4x4.

```
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2D(16, (3, 3), activation='relu')(x)
x = layers.UpSampling2D((2, 2))(x)
decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)
```

Реалізація декодера

У випадку с декодером, в цьому коді навпаки відбувається збільшення розмірності даних, тобто генерація даних з латентого-простору.

```
autoencoder = keras.Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

Навчання мережі

Висновки до розділу 3

В даному розділі проведено порівняльний аналіз популярних інструментів побудови нейронних мереж. В ході порівняння було обрано бібліотеку Keras, яка має досить зручний і мінімальний інтерфейс, який дуже вдало підходить для реалізації мережі. Як бекенд для Keras було обрано популярну бібліотеку Tensorflow.

Було описано ключові моменти архітектури автокодувальників. Виявлено недоліки і переваги таких мереж.

РОЗДІЛ 4. РЕЗУЛЬТАТИ І ВИСНОВКИ

4.1 Висновки

На жаль, не існує метрики, яка б тверезо могла б оцінити якість реставрації зображення і явно вказати який метод є найкращим. Найчастіше, для оцінки якості таких методів, використовується суб'єктивне сприйняття, тому головна складова оцінки - це візуальна схожість виправленого зображення на оригінал.

На малюнку 2 представлені результати роботи різних методів автоматичної реставрації зображення. Для прискорення перевірки гіпотези і спрощення навчання мережі, використовуються невеликі зображення 32x32 пікселя з дата-сету CIFAR10.

Кінцеве зображення яхти і автомобіля (1 і 2 зображення) має досить велику схожість з оригінальною картинкою. Як було сказано вище, алгоритми статичного аналізу відмінно справляється у випадках, коли явно можна визначити межі об'єктів. Саме це можна спостерігати на кінцевих зображеннях. При наявності явних кордонів кольору і можливості монотонного заповнення цим кольором пропущеного простору, відреставроване зображення виглядає максимально схожим на оригінал. Що стосується автоендеров, то вони теж добре впоралися із завданням реставрації, маски були успішно видалені. Слід зауважити, що автоенкодери має один дефект, який спостерігається абсолютно на всій підсумкової вибірці, а саме змінити тон кольору зображення. Можна помітити, що зображення, яке було створене автоенкодером, має жовтий відтінок. Це пов'язано з тим, що завжди присутні втрати при відновленні зображення з латентного простору.

Оригінальне зображення	Пошкоджене зображення	Метод Навьє-Стокса	Fast-Marching Метод	Autoencoder
				
				
				

Таблиця 6 – Порівняння методів автоматичної реставрації зображення

З реставрацією собаки (3 зображення) не все так однозначно, як в попередніх випадках. Можна помітити, що маска була успішно видалена, але от якість зображень, які були отримані в результаті використання статичних методів, залишає бажати кращого. Явно можна спостерігати силуети тварини, але якість деталей, для реставрації яких треба розуміти контекст, невисока. Абсолютно відсутня ніс і візуальні кордону голови і тулуба тварини. У випадку з автоенкодером, то все там якість деталей значно вище. Так само присутній ефект зі зміною кольору, але майже повністю відсутні артефакти, які були описані вище.

Методи статичного аналізу та автоекодери мають свої переваги і недоліки.

Сильною стороною статичних алгоритмів є редагування зображень, які не наповнені деталями. Так само для роботи таких алгоритмів не треба витрачати час на навчання мереж і підбору дата-сетів, що значно спрощує їх використання. Так як алгоритм завжди чітко знає область, яку необхідно реставрувати, тому інша частина зображення не втрачає в якості і передачі кольору. Недоліками таких методів є обов'язкове точне розуміння реставрується області, тобто алгоритм не може працювати повністю автоматично, без участі людини. Через те, що алгоритм намагається аналізувати навколишні пікселі, то не виникає висока ймовірність втрати деталей.

У автоенкодеров абсолютно відсутні такі недоліки. Наприклад, такий підхід може працювати абсолютно автоматично, немає необхідності явного вказування відновлюваної області. Через наявність згорткових нейронний мереж, метод має більш точне розуміння контексту, що значно підвищує якість фінальних зображень. Але, не дивлячись на більш високу деталізацію, внаслідок втрат, можуть, з'являється інші і артефакти, такі як незначна зміна кольору.

На основі порівняння зображень, які було відреставровано автоекодером, і оригінальним можна зробити висновок, що використання нейронних мереж для вирішення подібних завдань абсолютно виправдано і майже не поступається іншим підходам, а в деяких випадках навіть перевершує.

У таблиці представлені результати роботи автоекодера. Таблиця складається з 4 стовпців, кожен з них містить зображення. Перший стовпчик - це зображення з штучно створеної маскою, яка була нанесена на оригінальне зображення з CIFAR10 дата-сету. При огляді результатів спостерігається, що все маски були успішно видалені, ніде не залишилося слідів пошкодження.

Як було сказано вище, автоенкодери при відновленні зображень завжди втрачають в якості. Саме цю втрату інформації можна спостерігати на підсумкових зображеннях. Самим явною ознакою втрат є зміна колірних відтінків зображень. Сильнішого зміни зазнали більш яскраві кольори (червоний, жовтий), в той же час більш темні кольори (синій, чорний, фіолетовий) залишилися майже без візуальних змін. На малюнку 22 так само представлені результати робота автоенкодеров, але з абсолютно іншою маскою. В цьому випадку міський є не випадково згенеровані лінії, а статичний квадрат. При такому підборі дефектів автоекодери теж впоралися зі своїм завданням, але з куди більш помітними візуальними артефактами. Наприклад, у випадку з машиною, маска майже не відвернулась, тільки стала менш помітною. На малюнку 23 зображені результати роботи одного з алгоритмів статичного аналізу, а саме Fast Marching. При огляді результатів стає видно, що якість результуючих зображень куди нижче. Так як метод заснований на аналізі пікселів, які оточують область для реставрації. У цьому випадку, коли маска займається досить велика кількість місця, то інформації оточуючих пікселів недостатньо. Різного роду артефакти з'являються, коли для відновлення видаленої області, необхідно володіти розуміємо контексту. Різниця між походами чітко спостерігається у випадках, коли редагується зображення собаки, наприклад. Нейронна мережа навчалася на великій вибірці фотографій, які включали в себе і зображень собак. Мережа навчилася створювати таке латентне-простір, з якого можна легко відновити зображення тварини. У випадку ж з Fast Marching методу, то при відновлення області втрачається значна кількість деталей. Наприклад, при реставрації тієї ж собаки, то в процесі роботи методу, у тварини пропав ніс. А ось при роботі з більш тонкими лініями (таблиця номер 24) Fast Marching справляється куди краще. У випадках з тонкими лініями, то інформації в сусідніх пікселях досить, щоб визначити подальший розподіл кольору. При наявності більшої кількості ліній і збільшення їх ширини, то інформації стає недостатньо, що призводить до появи розмитості і зникнення деталей.

При використанні методу Нав'є-Стокса підсумкові зображення виходять більш деталізовані, а межі переходу між квітами більш плавними.

4.2 Що можна поліпшити?

Очевидно, що з картинок 32x32 можна витягти мало практичної цінності. Як мінімум через те, що у звичайних користувачів немає потрібні працювати з настільки маленькими зображеннями. Одним з пусей подальшого розвитку нейронної мережі, може стати її масштабування. Збільшення розмірів вихідних зображень, внесення інкрементальних поліпшень в структуру автокодувальників.

Автокодувальники - це по суті є алгоритмом для стиснення даних. Одна з проблем такі алгоритмом є те, що стиснення відбувається з втратами. При роботі такі втрати даних можуть відбиватися на якості деталей і несуттєве зміна кольору. Якщо проблема з деталями може зважитися ускладненнями мережі, то з кольором не все так просто. Одним з популярних методів, які зменшують візуальне колірне розходження між зображеннями - це використання ще однієї нейронної мережі, основною метою якої буде відновлення колірного рівня в зображенні.

.

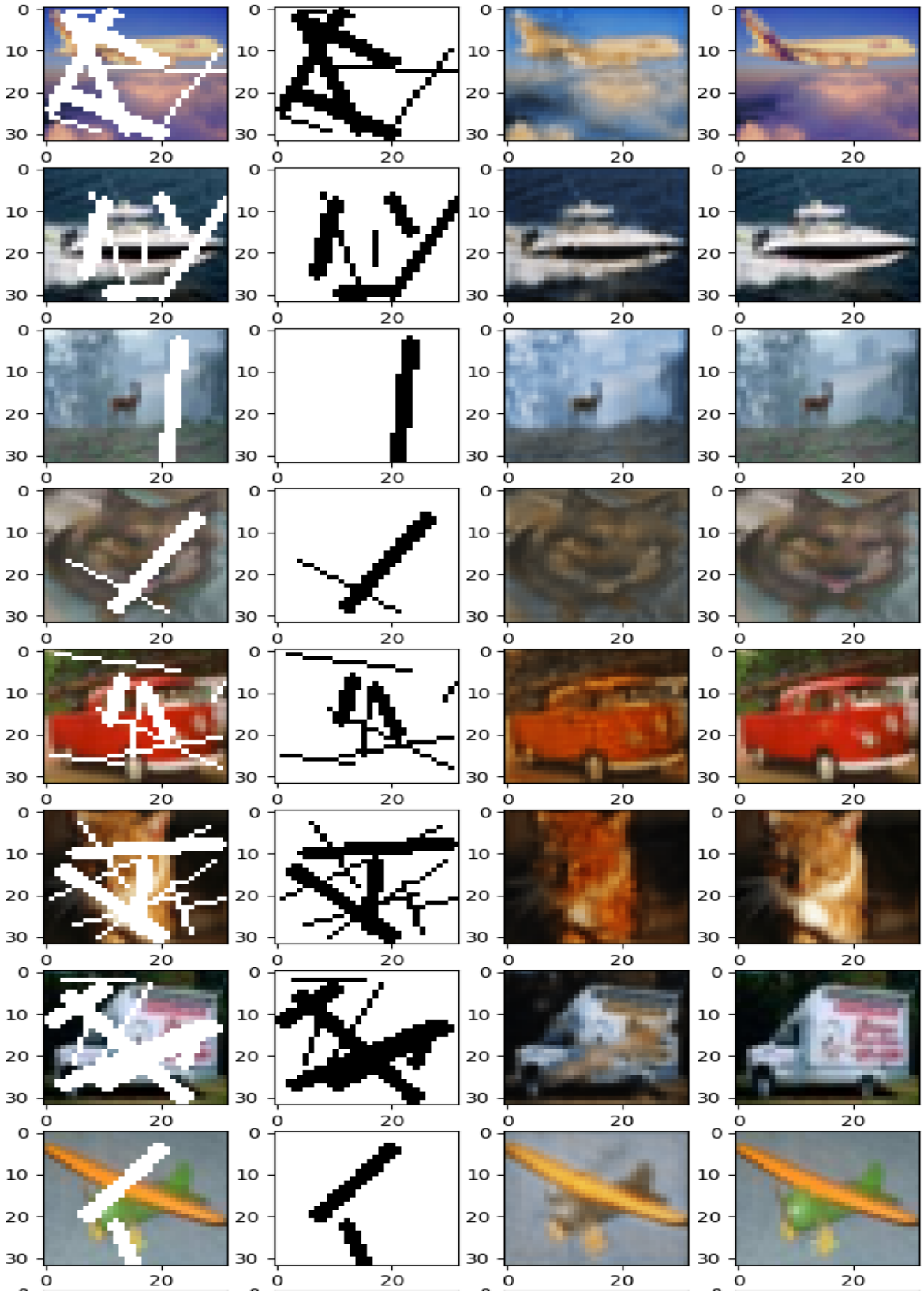


Рисунок 24 – результат роботи автокодувальників з випадковими масками

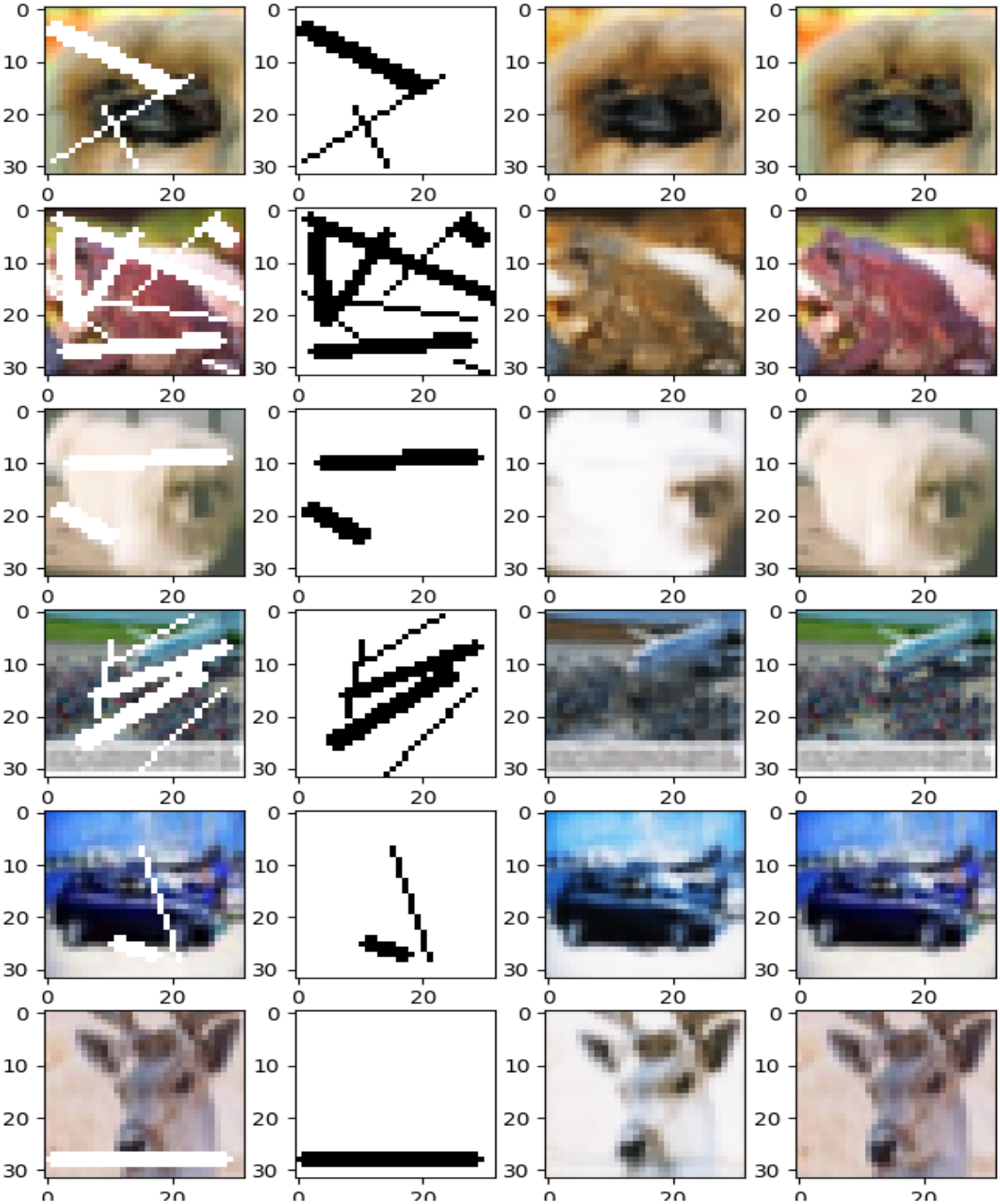


Рисунок 25 – Результати роботи Fast Marching методу з випадковими масками

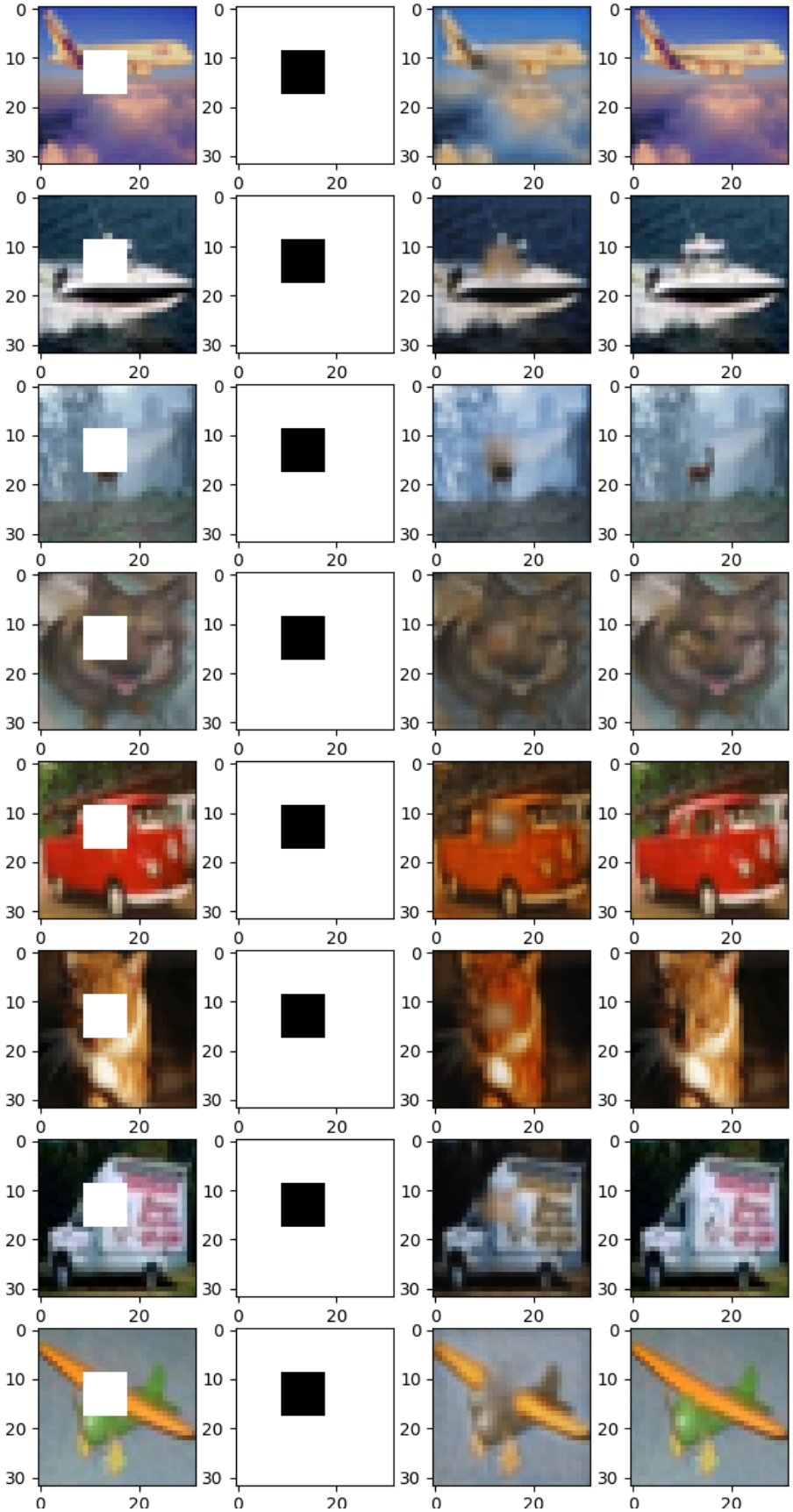


Рисунок 26 – Результат роботи автокодувальника зі статичною маскою

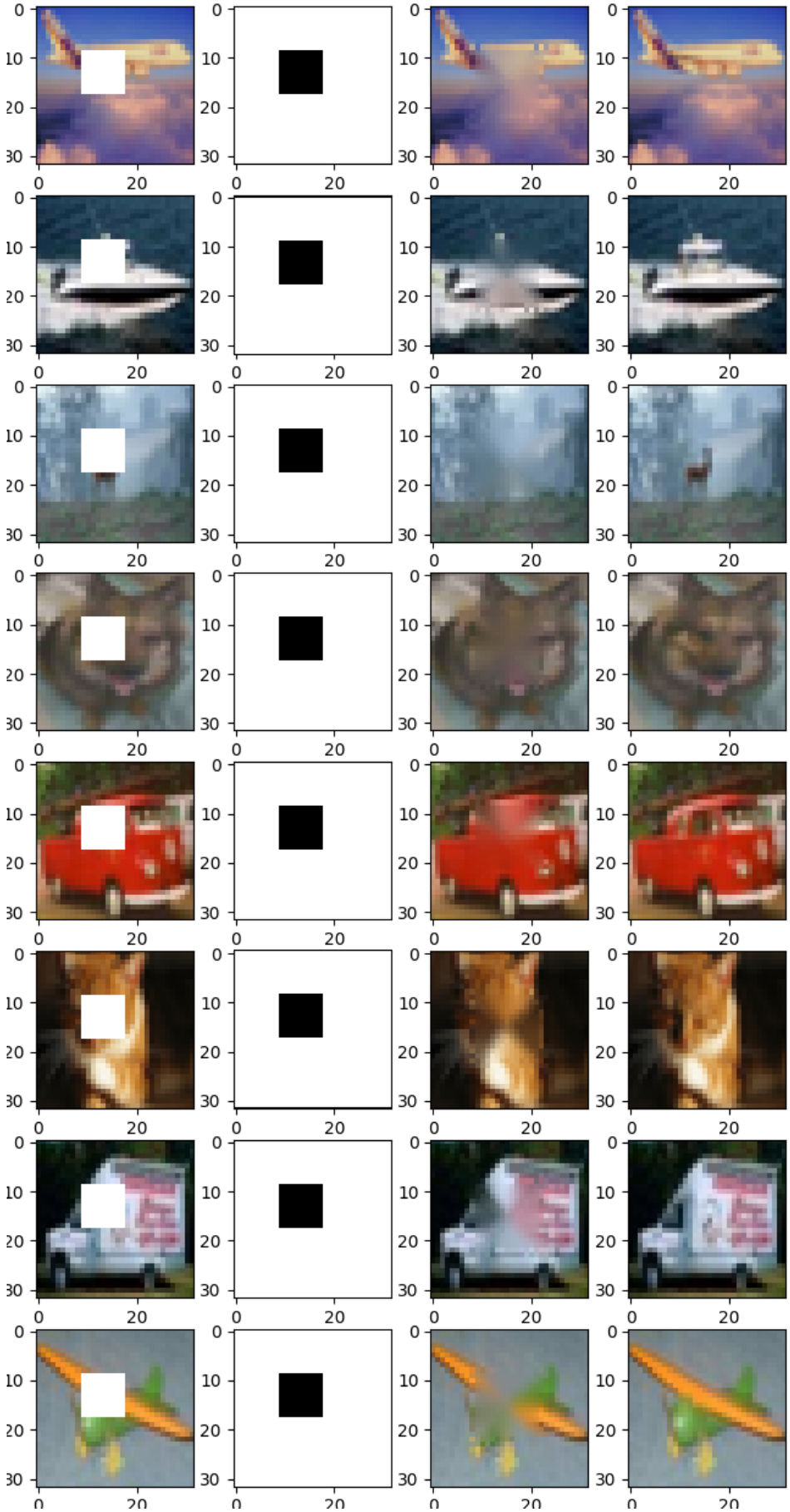


Рисунок 27 – Результат роботи Fast Marching зі статичною маскою

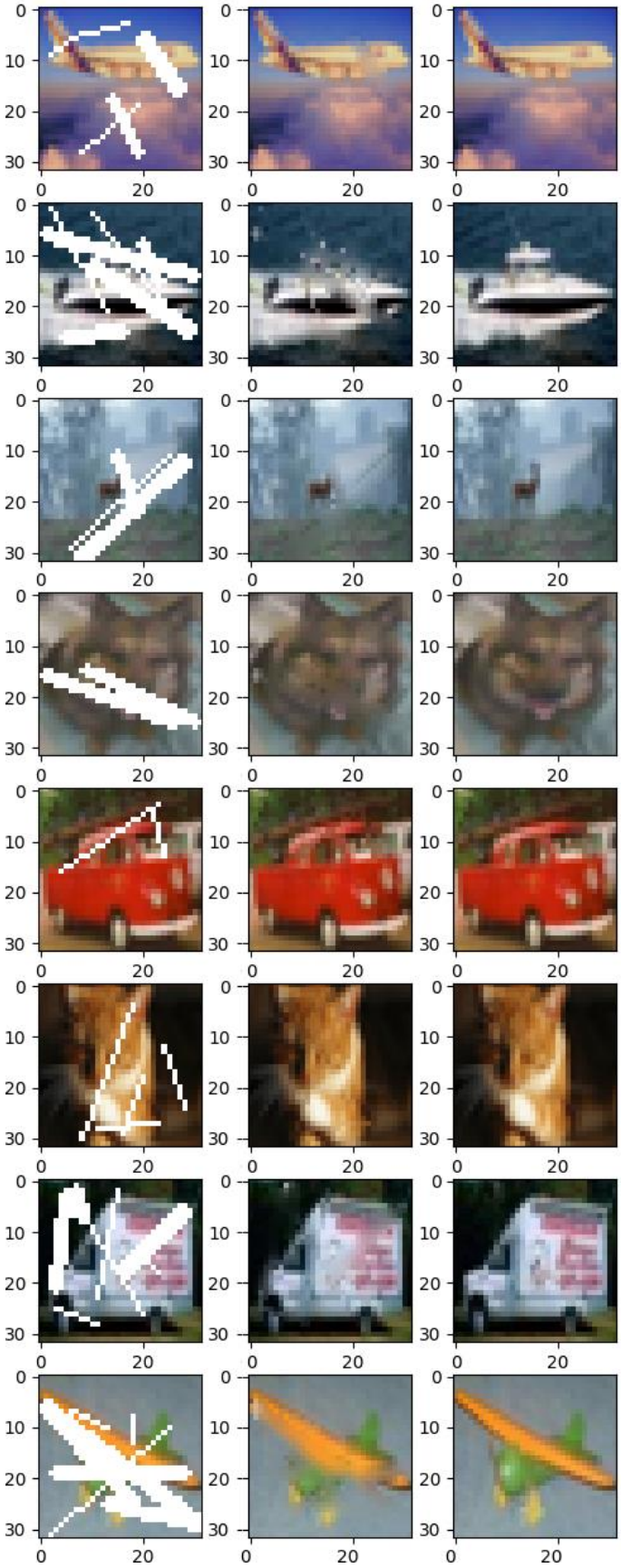


Рисунок 28 – Результат роботи Fast Marching метода з випадковою маскою

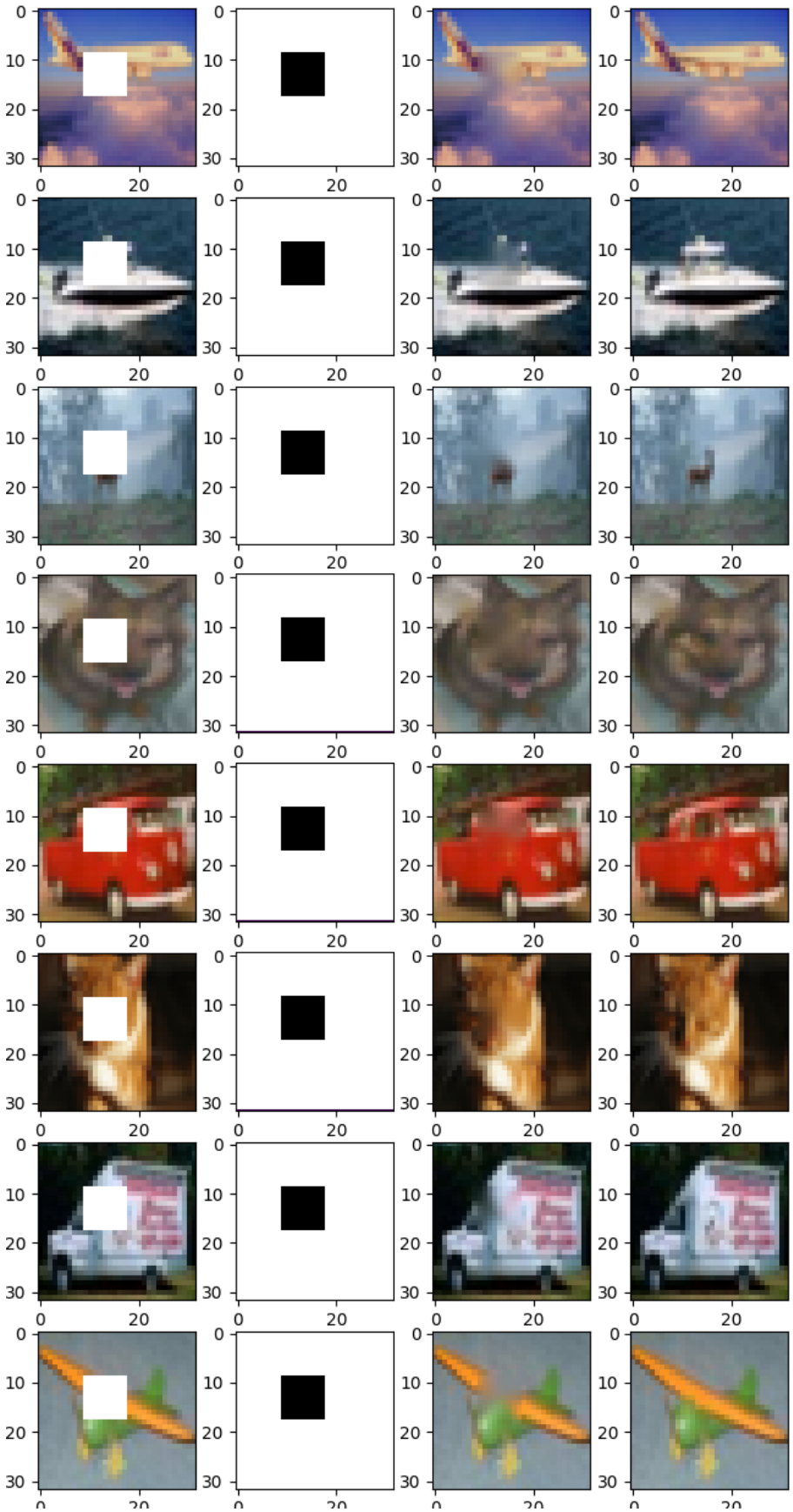


Рисунок 29 – Результат роботи методу Навьє-Стокса зі статичними масками

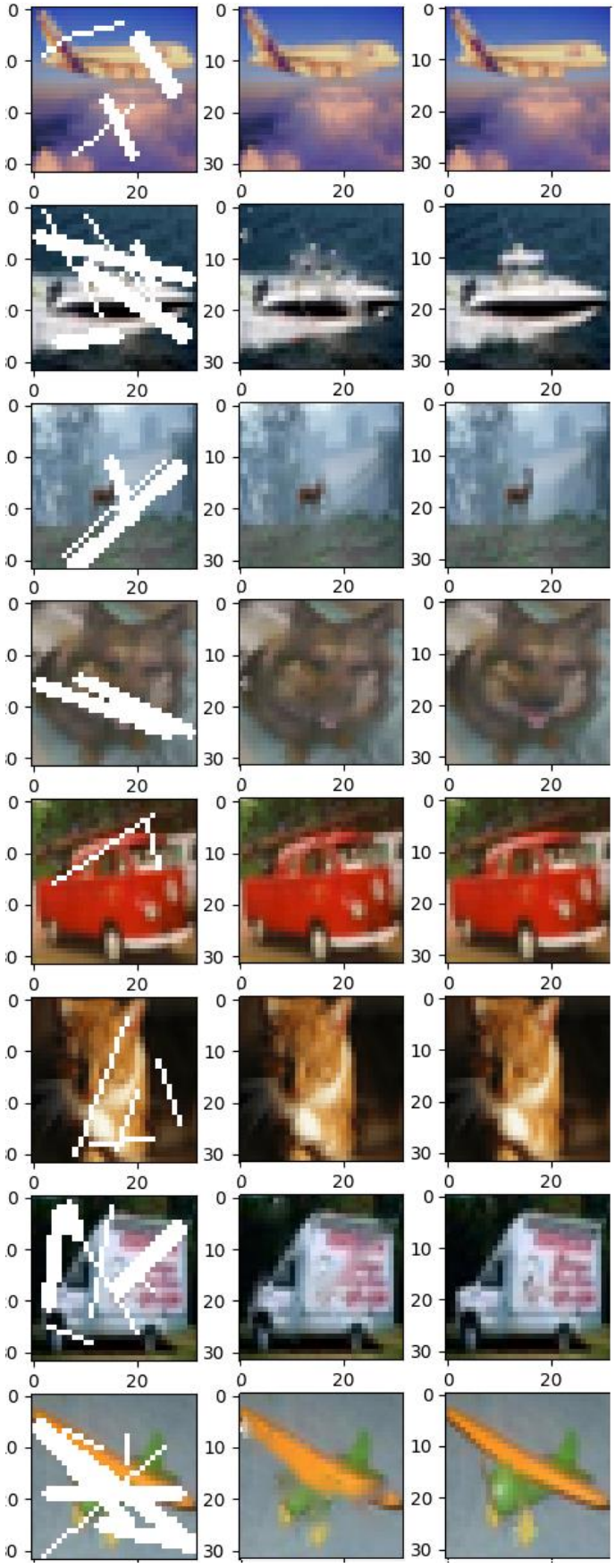


Рисунок 30 – Результат роботи методу Навьє-Стокса з випадковою маскою

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В. Штеньович. Вступ до машинного навчання / В. Штеньович, А. Білоус // Режим доступу: https://dou.ua/lenta/articles/introduction-machine-learning-1
2. A. Thakur introduction-to-image-inpainting-with-deep-learning // Режим доступу: https://www.wandb.com/articles/introduction-to-image-inpainting-with-deep-learning
3. Keras // Режим доступу: https://keras.io/
4. Metrics to evaluate your semantic segmentation model // Режим доступу: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2
5. Autoencoders // Режим доступу: https://towardsdatascience.com/tagged/autoencoder
6. Introduction to autoencoders // Режим доступу: https://www.jeremyjordan.me/autoencoders/
7. NVIDIA Corporation // Режим доступу https://arxiv.org/pdf/1804.07723.pdf
8. Автоэнкодеры в Keras. // Режим доступу https://habr.com/ru/post/331382/
9. Applied-deep-learning-part-3-autoencoders // Режим доступу https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders
10. Auto-encoder-what-is-it // Режим доступу https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726
11. Autoencoders // Режим доступу https://www.deeplearningbook.org/contents/autoencoders.html
12. Autoencoders. TensorFlow // Режим доступу https://www.tensorflow.org/tutorials/generative/autoencoder